USER'S GUIDE FOR THE iSBC 957B™ iAPX 86, 88 INTERFACE AND EXECUTION PACKAGE

Order Number: 143979-002

Copyright © 1981 Intel Corporation
Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051

REV.	REVISION HISTORY	PRINT DATE
-001	This product is based on the iSBC 957A product but it contains modifications making the iSBC 957B product more useful for the iRMX 86, 88 operating system.	9/81
-002	This republication is to correct minor errors.	10/81

Additional copies of this manual or other Intel literature may be obtained from:

Literature Department
Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

The information in this document is subject to change without notice.

Intel Corporation makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Intel Corporation assumes no responsibility for any errors that may appear in this document. Intel Corporation makes no commitment to update nor to keep current the information contained in this document.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied.

Intel software products are copyrighted by and shall remain the property of Intel Corporation. Use, duplication or disclosure is subject to restrictions stated in Intel's software license, or as defined in ASPR 7-104.9(a)(9).

No part of this document may be copied or reproduced in any form or by any means without the prior written consent of Intel Corporation.

The following are trademarks of Intel Corporation and its affiliates and may be used only to identify Intel products:

BXP	Intelevision	Micromap
CREDIT	Intellec	Multibus
i	iRMX	Multimodule
ICE	iSBC	Plug-A-Bubble
iCS	iSBX	PROMPT
im	Library Manager	Promware
INSITE	MCS	RMX/80
Intel	Megachassis	System 2000
Intel	Micromainframe	UPI
		μScope

and the combination of ICE, iCS, iRMX, iSBC, iSBX, MCS, or RMX and a numerical suffix.

A662/782/3K DD

PREFACE

This manual provides general information, interfacing instructions and programming information for the Intel iSBC 957B iAPX 86, 88 Interface and Execution Package. Additional information is available in the following documents:

Manual	Number
iSBC 86/12A Single Board Computer Hardware Reference Manual	9803074
iSBC 86/05 Single Board Computer Hardware Reference Manual	143153
iSBC 88/40 Measurement and Control Computer Hardware Reference Manual	142978
iSBC 88/25 Single Board Computer Hardware Reference Manual	143825
iSBC 337 Multimodule Numeric Data Processor Hardware REference Manual	142887
iSBX 351 Serial Multimodule Board Hardware Reference Manual	9803190
8086 Family User's Manual	9800722
8086 Family User's Manual Numerics Supplement	121586
ISIS-II User's Guide	9800306
8086/8087/8088 Assembly Language Manual for 8086-Based Development Systems	121627
8086/8087/8088 Macro Assembler Operating Instructions for 8086-Based Development Systems	121628
iAPX 86, 88 Family Utilities User's Guide for 8086-Based Development Systems	121616
PL/M-86 User's Guide for 8086-Based Development Systems	121636
iRMX 86™ Configuration Guide	9803126
iRMX 86™ Loader Reference Manual	143318
iRMX 88™ Reference Manual	142232
iRMX 88™ Interactive Configuration Utilites	142603
Application Note - Getting Started With the Numeric Data Processor	AP-113

CONTENTS

PAGE

CHAPTER 1
GENERAL INFORMATION
Introduction..... 1-1
Description..... 1-1

CHAPTER 2
INSTALLATION
Equipment Supplied..... 2-1
General Hardware Configuration..... 2-1
Serial Interfacing and Board Jumper Configuration..... 2-1
iSBC 86/12A Board Jumper and Switch Settings..... 2-2
iSBC 86/05 Board Jumper and Switch Settings..... 2-2
iSBC 88/40 Board Jumper and Swtich Settings..... 2-4
iSBC 88/25 Board Jumper and Switch Settings..... 2-6
iAPX 86, 88 Monitor Program..... 2-7
iSBC 86/12A Board..... 2-7
iSBC 86/05 Board..... 2-7
iSBC 88/40 Board..... 2-7
iSBC 88/25 Board..... 2-7
Intellec System Jumpers..... 2-8
Cabling For Serial System Interfacing..... 2-8
Intellec Series II Model 210..... 2-8
Intellec Series II Model 220/230..... 2-8
Intellec 800..... 2-8
Parallel Interfacing..... 2-11
Cabling For Parallel System Interfacing..... 2-12
Intellec Series II Model 210..... 2-12
Intellec Series II Model 220/230..... 2-12
Intellec 800..... 2-13
Interfacing Without an Intellec System..... 2-14

CHAPTER 3
OPERATION
Start-Up Procedure..... 3-1
Command Structure..... 3-2
Byte and Word Variables..... 3-2
Numeric (Real, Integer and BCD) Variables..... 3-3
Address Specification..... 3-7
Multiple Commands On A Single Line..... 3-7
iAPX 86 and iAPX 88 CPU Registers..... 3-8
NPX Registers..... 3-9
Errors..... 3-9
Entering Commands..... 3-10
Command Descriptions..... 3-11
Load Absolute Object File (L)..... 3-13
Go (G)..... 3-14
Load and Go (R)..... 3-15

CONTENTS (continued)

PAGE

CHAPTER 3 (continued)
Command Descriptions
- Upload (T) 3-16
- Single Step (N) 3-17
- Examine/Modify Register (X) 3-18

iAPX 86 and iAPX 88 Registers 3-19
NPX Registers and Stack Registers 3-20
- Display Memory (D) 3-22
- Substitute (S) 3-25
- Move (M) 3-27
- Find (F) 3-28
- Compare (C) 3-29
- Port Input (I) 3-29
- Port Output (O) 3-30
- Print (P) 3-31
- Exit (E) 3-33
- Comment (*) 3-33
- Bootstrap (B) 3-33

CHAPTER 4
SYSTEM I/O ROUTINES
Libraries for PL/M-86 Cases 4-1
- Open Routine 4-2
- Read Routine 4-4
- Write Routine 4-5
- Seek Routine 4-6
- Rescan Routine 4-8
- Close Routine 4-8
- Delete Routine 4-9
- Rename Routine 4-10
- Attrib Routine 4-11
- Load Routine 4-12
- Error Routine 4-13
- CI Routine 4-14
- CO Routine 4-14
- ISIS Routine 4-15
- Exit Routine 4-17

CHAPTER 5
PROGRAMMING INFORMATION
Memory Organization 5-1
Special Considerations When Stepping 5-2
iAPX 86, 88 CPU Register Initialization 5-3
USART Initialization 5-3
- Interrupt Servicing 5-4

Instruction Breakpoints 5-5
NPX Initialization 5-5
EEPROM Timer Initialization 5-5

PAGE

CHAPTER 6
CONFIGURING THE iAPX 86, 88 MONITOR
Configuration Source File Contents..... 6-2
The Monitor Configuration Macros..... 6-3
The CPU Macro..... 6-3
The MAX_BAUD_RATE_COUNT Macro..... 6-3
The BAUD_RATE Macro..... 6-5
The BAUD_RATE_TIMER Macro..... 6-6
The EXTRA_TIMER Macro..... 6-7
The SERIAL_PORT Macro..... 6-8
The PARALLEL_PORT Macro..... 6-9
The INTERRUPT_CONTROLLER Macro..... 6-9
The NPX Macro..... 6-10
Bootstrap Loader Option..... 6-11
The BOOTSTRAP Macro..... 6-11
The DEVICE Macro..... 6-11
The END_BOOTSTRAP Macro..... 6-12
Sample Monitor Configurations..... 6-12
Monitor Configuration Submit Files..... 6-13
Default and Device Change Submit Files..... 6-14
NPX Support Submit Files..... 6-16
Bootstrap Loader Option Configuration Submit Files..... 6-13
Both NPX Support and Bootstrap Loader Support for the iSBC 88/40 Board..... 6-20

APPENDIX A
NUMBERED ISIS-II ERROR MESSAGES..... A-1

APPENDIX B
iAPX 86, 88 MONITOR ERROR MESSAGES..... B-1

APPENDIX C
EQUIPMENT SUPPLIED WITH THE iSBC 957B PACKAGE..... C-1

APPENDIX D
iSBC 901 and iSBC 902 Schematics..... D-1

FIGURES

2-1. iAPX 86, 88-Based Board to Intellec Series II Model 210 Serial Interface Cabling..... 2-9
2-2. iAPX 86, 88-Based Board to Intellec Series II Model 220/230 Serial Interface Cabling..... 2-10
2-3. iAPX 86, 88-Based Board to Intellec 800 Serial Interface Cabling..... 2-10

FIGURES (continued)

		PAGE
2-4.	iAPX 86, 88-Based Board to Intellec Series II Model 210 Parallel Interface Cabling	2-12
2-5.	iAPX 86, 88-Based Board to Intellec Series II Model 220/230 Parallel Interface Cabling	2-13
2-6.	iAPX 86, 88-Based Board to Intellec 800 Parallel Interface Cabling	2-13
2-7.	iAPX 86, 88-Based Board to CRT Console Device	2-14

TABLES

2-1.	Parallel Interfacing Jumpers	2-11
3-1.	NPX Data Types	3-4
3-2.	iAPX 86, 88 CPU Registers	3-8
3-3.	NPX Registers	3-9
3-4.	Summary of Loader and Monitor Commands	3-11
4-1.	Summary of System I/O Routines	4-1
6-1.	iAPX 86, 88 Monitor Configuration Source Files	6-1
6-2.	iAPX 86, 88 Monitor Configuration Submit Files	6-14
A-1.	Nonfatal Error Numbers Returned by System Calls	A-3
A-2.	Fatal Errors Issued by System Calls	A-3

CHAPTER 1. GENERAL INFORMATION

INTRODUCTION

The iSBC 957B iAPX 86, 88 Interface and Execution Package (referred to in this manual as the iSBC 957B package) consists of two parts, a loader program and an iAPX 86, 88 monitor. The loader, which executes in an Intel Intellec development system with at least 64K of RAM, has the capability of loading files either from the Intellec system to an iAPX 86, 88-based board (downloading) or in the reverse direction (uploading). The monitor, which resides in EPROM on an iAPX 86, 88-based board, supports both interactive commands and ISIS-like system I/O routines. The commands are useful during debugging and supplement the capabilities of the ICE-86 In-Circuit Emulator. The I/O routines can be used by programs running on an iAPX 86, 88-based board; they enable the program to access and use files in the Intellec system.

The iSBC 957B package is compatible with both the iSBC 86/12 and the iSBC 86/12A (refered to in this manual as only the iSBC 86/12A), the iSBC 86/05, the iSBC 88/40 and the iSBC 88/25 single board computers (SBC). Throughout this manual they will be collectively referred to as "iAPX 86, 88-based boards".

The iSBC 957B package supports the use of Electrically Erasable Programmable Read Only Memory (EEPROM) when the monitor is installed on an iAPX 86, 88-based board that has hardware to support EEPROM. Values in EEPROM may be changed or executable code may be loaded into EEPROM using the iSBC 957B package.

The iSBC 957B package supports the use of an 8087 Numeric Processor Extension installed on an iAPX 86, 88-based board.

The iSBC 957B package replaces the iSBC-957A Intellec-iSBC 86/12A Interface and Execution Package.

DESCRIPTION

The iSBC 957B package includes the following:

a. Loader software and monitor EPROM's.

b. Three cable assemblies.

c. Line driver/terminators used for parallel input and output.

d. Source modules, submit files and a library for configuring custom versions of the iAPX 86, 88 monitor.

Also supplied are four iSBC 901 Resistor Packs and four type 7437 line driver integrated circuits. These components, which are not used with the implementation of this package, are supplied for the user's own purpose; refer to paragraph 2-11 in the iSBC 86/12A SINGLE BOARD COMPUTER HARDWARE REFERENCE MANUAL.

The loader is provided on diskettes and operates under ISIS-II control. The cables, line drivers, and terminators support the different iAPX 86, 88-Intellec hardware configurations. The iAPX 86, 88 monitor configuration source modules, submit files and configuration library are provided on diskettes and require a Series-III Intellec development system to use.

Complete instructions for interfacing the iSBC 86/12A, the iSBC 86/05, the iSBC 88/40 and the iSBC 88/25 boards with each of the Intellec models are provided in Chapter 2 and operating information is given in Chapter 3. System I/O routines are presented in Chapter 4, with additional programming information being supplied in Chapter 5. Descriptions of how to configure custom versions of the iAPX 86, 88 monitor are given in Chapter 6. The appendices contain lists of error messages, a list of the equipment supplied with the iSBC 957B package and schematics of the iSBC 901 and iSBC 902 resistor packs.

CHAPTER 2. INSTALLATION

This chapter provides the necessary information to configure the iSBC 957B package to interface with an Intellec system. The iAPX 86, 88-based boards must be connected to a spare channel (serial or parallel) on the Intellec system. Therefore, the hardware requirement depends on the type (model) of Intellec system and the type of console device in use. The iAPX 86, 88-based board can also be attached in a standalone configuration.

If the serial port is used by the application program, the parallel port can be used for interfacing the iAPX 86, 88-based board to the Intellec system or vice versa.

EQUIPMENT SUPPLIED

A list of the equipment supplied with the iSBC 957B package is provided in appendix C.

GENERAL HARDWARE CONFIGURATION

The iAPX 86, 88-based boards should be installed in a chassis other than that of the Intellec system. The use of any iAPX 86, 88-based boards in the Intellec system chassis is not recommended and is not supported. The instructions to configure the serial I/O ports and the parallel I/O ports for the iSBC 86/12A, iSBC 86/05, iSBC 88/40 and iSBC 88/25 single board computers are given in the following sections. References to "default jumpers" or "default jumper configurations" refer to the jumpers installed at the factory that are on the iAPX 86 or 88 based board when it is delivered. All Intel single board computers are delivered with the Nonmaskable Interrupt pin grounded. Unless the Nonmaskable interrupt is used for an explicit purpose, this pin must remain grounded. Refer to the section in Chapter 5 on Instruction Breakpoints for information on how to use the Nonmaskable Interrupt with the monitor.

SERIAL INTERFACING AND BOARD JUMPER CONFIGURATION

If the parallel interface is needed by your application program, a serial interface may be used on an iAPX 86, 88-based board for communication with the Intellec development system. This interface requires the exclusive use of an 8251A USART. Included with the instructions for configuring the serial interface are instructions for other required hardware modifications.

iSBC 86/12A Board Jumper and Switch Settings

Except where noted in this chapter, the jumper and switch requirements for configuring the ROM/EPROM, Timer Input Frequency (Counter 2), and Serial I/O Port for the iSBC 86/12A board are the default jumper configurations. Refer to the iSBC 86/12A SINGLE BOARD COMPUTER HARDWARE REFERENCE MANUAL for details.

ROM/EPROM CONFIGURATION - The required jumper and switch settings for supporting the ROM/EPROM on the 86/12A board are as follows:

DEFAULT JUMPERS	SWITCH S1
E94-E96	8-9 (open) 7-10 (closed)
REMOVE JUMPER	**ADD JUMPER**
E97-E98	E97-E99

TIMER (COUNTER 2) INPUT FREQUENCY - The 8253 input frequency for counter 2 (8251A Baud Rate Clock) is 1.23 MHz. This is selected by the default jumper connection E54-E55.

SERIAL I/O PORT CONFIGURATION - The necessary serial I/O port jumpers are:

DEFAULT JUMPERS	ADD JUMPER
E39-E40 E42-E43 W1 (A-B) W2 (A-B) W3 (A-B)	E51-E52

TIME OUT OPTION - Inadvertent attempts to access non-existent memory can cause the 8086 CPU to enter an endless series of WAIT states. The default jumper connection E5-E6 will prevent such an occurrence.

iSBC 86/05 Board Jumper and Switch Settings

Except where noted in this chapter, the jumper requirements for configuring the ROM/EPROM, Timer Input Frequency (counter 2) and the Serial I/O port for the iSBC 86/05 board are the default jumper configurations. This will configure the iSBC 86/05 board to run at 8MHz with two WAIT states inserted for ROM/EPROM memory accesses. Refer to the iSBC 86/05 SINGLE BOARD COMPUTER HARDWARE REFERENCE MANUAL for details.

ROM/EPROM CONFIGURATION - The required jumper settings for supporting the ROM/EPROM on the iSBC 86/05 board are as follows:

REMOVE JUMPER	ADD JUMPER
U35, 2-13	U35, 6-9 210-211

TIMER (COUNTER 2) INPUT FREQUENCY - The 8253 input frequency for counter 2 (8251A Baud Rate Clock) is 1.23 MHz. This is selected by the default jumper connection E56-E57.

SERIAL I/O PORT CONFIGURATION - The necessary serial I/O port jumpers are:

DEFAULT JUMPERS	ADD JUMPER
E79-E80 E83-E84 E89-E90 E91-E92	E87-E88

TIME OUT OPTION - Inadvertent attempts to access non-existent memory can cause the 8086 CPU to enter an endless series of WAIT states. The default jumper connection, E14-E15, will prevent such an occurrence.

NONMASKABLE INTERRUPT - To ensure that the Nonmaskable Interrupt is protected against noise so it will not randomly interrupt the CPU, connect jumpers E26-E27.

WAIT STATE OPTIONS - The default jumper connection E6-E7 provides for two WAIT states to be inserted when reading from or writing to ROM/EPROM memory. This corresponds to the access time of the 2732 EPROMs supplied with the iSBC 957B package. If fewer wait states are desired, you must burn the iAPX 86, 88 monitor into EPROMs which have a faster access time and then must configure the WAIT state selector jumpers appropriately. See the iSBC 86/05 SINGLE BOARD COMPUTER HARDWARE REFERENCE MANUAL for details.

COMMON BUS REQUEST (CBRQ) OPTIONS - To allow boards in the system that do not have a bus arbiter on them to access the Multibus, the iSBC 86/05 CBRQ line must be grounded. Without this option, the iSBC 86/05 board will retain control of the Multibus until another bus arbiter requests the bus. This will prevent boards without a bus arbiter on them from gaining use of the bus. An example of a board needing this capability is the iSBC 204 single density floppy diskette controller performing DMA. Modify the jumpers as follows:

REMOVE JUMPER	ADD JUMPER
E183-E184	E184-E185

iSBC 88/40 Board Jumper and Switch Settings

Except where noted in this chapter, the jumper requirements for configuring the ROM/EPROM, Timer Input Frequency (counter 2) and the Serial I/O port for the iSBC 88/40 board are the default jumper configurations as defined in the iSBC 88/40 MEASUREMENT AND CONTROL COMPUTER HARDWARE REFERENCE MANUAL.

ROM/EPROM CONFIGURATION - The required jumper settings for supporting the ROM/EPROM on the iSBC 88/40 board when installing the default monitor or a monitor configured without NPX support are as follows:

REMOVE JUMPERS	ADD JUMPERS
E99-E100	E94-E99
E108-E109	E108-E111
E119-E121	E118-E119
	E128-E131

When installing a monitor with NPX support configured in, the required jumper settings for supporting ROM/EPROM are as follows:

REMOVE JUMPERS	ADD JUMPERS
E99-E100	E98-E99
E108-E109	E108-E111
E119-E121	E118-E119
	E128-E131

Also remove all jumpers which are required to support EEPROMs, listed under "EEPROM OPTION", if they have been added.

SERIAL I/O PORT CONFIGURATION - The iSBC 88/40 board does not include an on-board USART. If serial I/O is desired, an iSBX 351 multimodule must be added to the iSBC 88/40 board. The default (Intel-supplied) iSBC 88/40 version of the iAPX 86, 88 monitor supports the iSBX 351 multimodule installed in the SBX B (J4) connector. This default configuration will not allow both the serial port and parallel port to be configured for the iSBC 88/40 board at the same time due to conflicting hardward requirements. One or the other, but not both, may be used. If desired, the SBX connctor for the iSBX 351 multimodule may be reassigned. See the section on configuration for details.

Follow the instructions in the iSBC 88/40 HARDWARE REFERENCE MANUAL for mounting the iSBX 351 multimodule. The iSBX 351 includes an 8251A USART and an 8253 Timer. The required jumper settings for the iSBX 351 are the following default jumper configurations:

DEFAULT JUMPERS

E3-E4
E5-E6
E9-E10
E11-E13
E12-E14
E17-E18
E19-E20
E27-E28
E29-E30
E37-E38

In addition, a 16 pin header must be wired as follows:

FROM PIN	TO PIN
pin 1	pin 16
pin 2	pin 15
pin 3	pin 13
pin 4	pin 14
pin 5	pin 6
pin 7	pin 10
pin 8	pin 9
pin 11	pin 12

Install this header in socket U6 on the iSBX 351.

This configuration will provide counter 2 of the 8253 Timer with an input clock frequency of 1.23 MHz.

TIMEOUT OPTION - Inadvertent attempts to access non-existent memory can cause the 8088 CPU to enter an endless series of WAIT states. The default jumper connection E267-E268 will prevent such an occurrence.

EEPROM OPTION - The default (Intel-supplied) iAPX 86, 88 monitor, configured for the iSBC 88/40 board, supports the use of a 2816 EEPROM installed in Socket U76. This default version of the monitor will use counter 2 of the on-board timer, in mode 1, to generate a programming pulse for writing to EEPROM. The required jumpers for this purpose are as follows:

ADD JUMPERS

E33-E35
E34-E36
E37-E38
E83-E85

iSBC 88/25 Board Jumper and Switch Settings

Except where noted in this chapter, the jumper requirements for configuring the ROM/EPROM, Timer Input Frequency (counter 2) and the Serial I/O Port for the iSBC 88/25 board are the default jumper configurations. This will configure the iSBC 88/25 board to run at 5 MHz with one WAIT state inserted for ROM/EPROM memory accesses. Refer to the iSBC 88/25 SINGLE BOARD COMPUTER HARDWARE REFERENCE MANUAL for details.

ROM/EPROM CONFIGURATION - The required jumper settings for supporting the ROM/EPROM on the iSBC 88/25 board are as follows:

REMOVE JUMPER	ADD JUMPER
J6 2-13	J6 6-9
	E90-E91

TIMER (COUNTER 2) INPUT FREQUENCY - The 8253 PIT input frequency for counter 2 (8251A Baud Rate Clock) is 1.23 MHz. This is selected by the default jumper connection E53-E54.

SERIAL I/O PORT CONFIGURATION - The necessary serial I/O port jumpers are:

DEFAULT JUMPERS	ADD JUMPER
E76-E77	E84-E85
E80-E81	
E86-E87	
E88-E89	

TIME OUT OPTION - Inadvertent attempts to access non-existent memory can cause the 8088 CPU to enter an endless series of WAIT states. The default jumper connection, E13-E14, will prevent such an occurrence.

iAPX 86, 88 MONITOR PROGRAM

iSBC 86/12A BOARD

The monitor program configured for the iSBC 86/12A board resides in four 2732 EPROMs. These EPROM's are supplied in the iSBC 957B package. Install these devices in the iSBC 86/12A board as follows:

PART NO.	SOCKET
143780-001	A28
143781-001	A29
143782-001	A46
143783-001	A47

iSBC 86/05 BOARD

The monitor program configured for the iSBC 86/05 board resides in four 2732 EPROMs. These are the same EPROMs as the iSBC 86/12A board uses and are supplied in the iSBC 957B package. Install these devices on the iSBC 86/05 board as follows:

PART NO.	SOCKET
143780-001	U66
143781-001	U67
143782-001	U33
143783-001	U34

iSBC 88/40 BOARD

The monitor program configurd for the iSBC 88/40 board resides in three 2732 EPROMs. These EPROMs are supplied in the iSBC 957B package. Install these devices on the iSBC 88/40 board as follows:

PART NO.	SOCKET
143784-001	U38
143785-001	U77
143786-001	U39

iSBC 88/25 BOARD

The iSBC 957B package includes no version of the monitor program configured for the iSBC 88/25 board. However, a submit file, CF8825.CSD, is provided on the diskettes supplied in the iSBC 957B package to configure the monitor for the iSBC 88/25 board. Refer to Chapter 6 for a description of how to use this submit file.

INTELLEC SYSTEM JUMPERS

Ensure that all the I/O ports (channels) of the Intellec system that are to be connected to the iAPX 86, 88-based board containing the iAPX 86, 88 monitor have all the default jumpers connected. Refer to the applicable Intellec system manual for that jumper configuration. During its initialization sequence, the loader reprograms the Intellec USART that connects to the iAPX 86, 88-based board for 9600 baud.

CABLING FOR SERIAL SYSTEM INTERFACING

The following paragraphs describe the interfacing required for serial communication between iAPX 86, 88-based boards and the various Intellec systems.

Intellec Series II Model 210

The Intellec Series II Model 210 must be upgraded with disk drives in order to use it with the iSBC 957B package.

iAPX 86, 88-based boards interface to the Serial port 1 of the Intellec Series II Model 210. Connect the cables as shown in Figure 2-1. Secure the RS232C connector to the Intellec chassis using two 4-40 panhead screws.

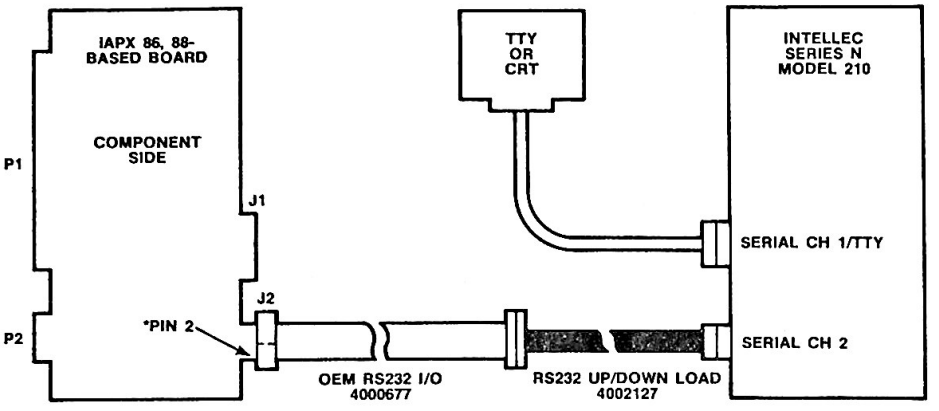
Intellec Series II Model 220/230

iAPX 86, 88-based boards interface to Serial port 1 of the Intellec Series II Model 220 or Model 230. Connect the cables as shown in Figure 2-2. Secure the RS232C connector to the Intellec chassis using two 4-40 panhead screws.

Intellec 800

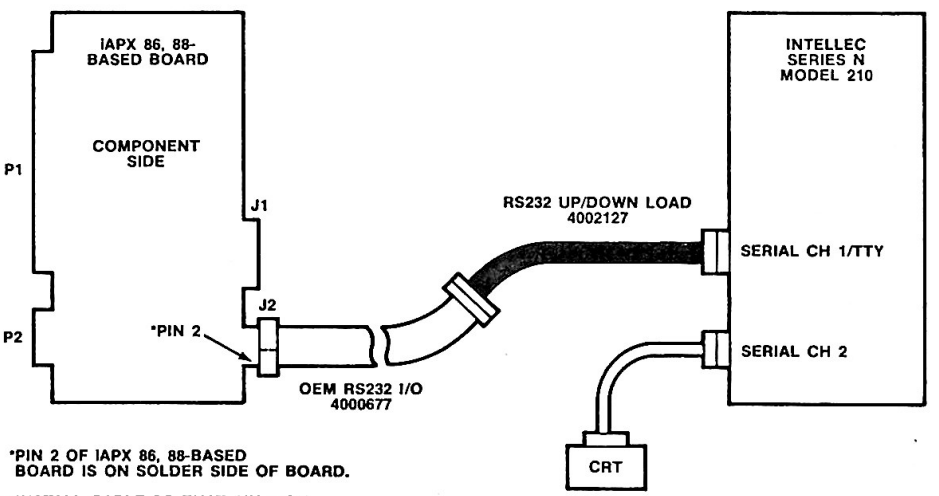
iAPX 86, 88-based boards can be interfaced to the CRT channel of the Intellec 800 when the Intellec console is a teletypewriter.

After the iAPX 86, 88-based board is configured, connect the cables as shown in Figure 2-3. Secure the RS232C connector to the Intellec chassis using two 4-40 panhead screws.



*PIN 2 OF iAPX 86, 88-BASED BOARD IS ON SOLDER SIDE OF BOARD.

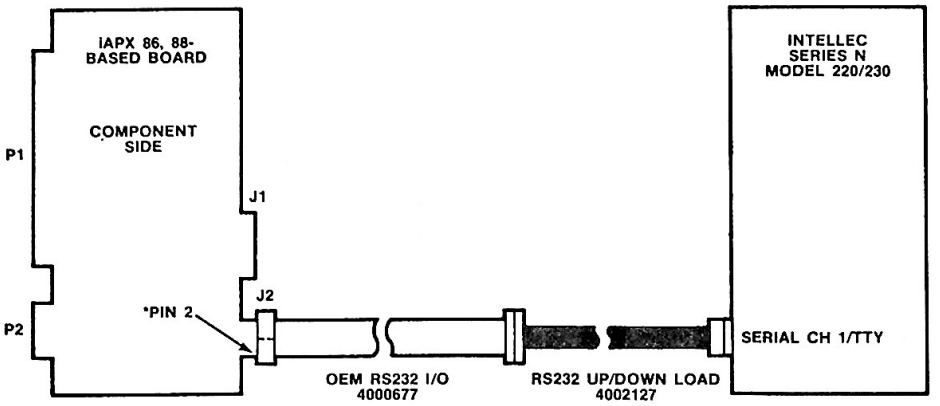
**INSTALL CABLE SO THAT PIN 1 ON CABLE CONNECTOR MATES WITH PIN 2 ON iAPX 86, 88-BASED BOARD EDGE CONNECTOR.



*PIN 2 OF iAPX 86, 88-BASED BOARD IS ON SOLDER SIDE OF BOARD.

**INSTALL CABLE SO THAT PIN 1 ON CABLE CONNECTOR MATES WITH PIN 2 ON iAPX 86, 88-BASED BOARD EDGE CONNECTOR.

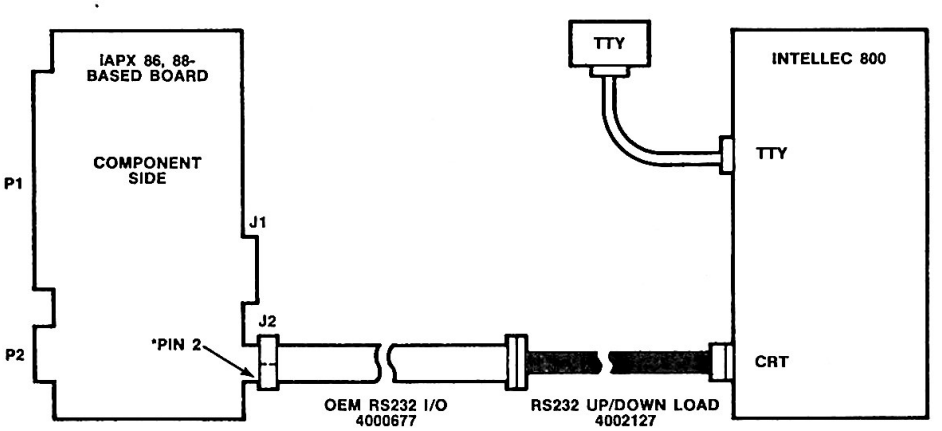
Figure 2-1. iAPX 86, 88-Based Board to Intellec Series II Model 210 Serial Interfacing Cabling



*PIN 2 OF iAPX 86, 88-BASED BOARD IS ON SOLDER SIDE OF BOARD.

**INSTALL CABLE SO THAT PIN 1 ON CABLE CONNECTOR MATES WITH PIN 2 ON iAPX 86, 88-BASED BOARD EDGE CONNECTOR.

Fig 2-2. iAPX 86, 88-Based Board to Intellec Series II Model 220/230 Serial Interface Cabling



*PIN 2 OF iAPX 86, 88-BASED BOARD IS ON SOLDER SIDE OF BOARD.

**INSTALL CABLE SO THAT PIN 1 ON CABLE CONNECTOR MATES WITH PIN 2 ON iAPX 86, 88-BASED BOARD EDGE CONNECTOR.

Fig. 2-3. iAPX 86, 88-Based Board to Intellec 800 Serial Interface Cabling

PARALLEL INTERFACING

If the serial interface is needed by your program or if large programs need to be loaded quickly, a parallel interface may be used on an iAPX 86, 88-based board for communication with the Intellec development system. This interface requires the exclusive use of an 8255A Programmable Peripheral Interface (PPI) on the processor board.

The iAPX 86, 88-based boards supported by the iSBC 957B package all require modification of the default PPI jumper configuration to use the parallel interface. Install the components and make the changes to the default jumpers as described in Table 2-1 for the type of iAPX 86, 88-based board being used.

Table 2-1. Parallel Interfacing Jumpers

	iSBC 86/12A	iSBC 86/05	iSBC 88/40	iSBC 88/25
iSBC 902 R-Packs	A10, A12, A13	U8, U10, U11	U2, U4, U7	U8, U10, U11
Status Adapter	All	U9	U3	U9
Remove Jumpers	E13-E14 E19-E20 E21-E25 E26-E27 E30-E31 E32-E33	E28-E29 E38-E42 E39-E43 E40-E44 E46-E50 E48-E52 E49-E53	E39-E41 E51-E52 E55-E56 E59-E60 E63-E64 E65-E66	E26-E27 E37-E41 E35-E39 E43-E47 E45-E49 E46-E50
Add Jumpers	E13-E27 E14-E30 E18-E31 E20-E33 E25-E31	E28-E48 E38-E49 E39-E43 E39-E48 E40-E52 E44-E46	E41-E63 E51-E64 E52-E59 E53-E63 E55-E65	E26-E45 E26-E36 E35-E46 E37-E49 E41-E43
Default Jumpers	E15-E16 E17-E18 E28-E29	E39-E43 E41-E45 E47-E51	E53-E54 E57-E58 E61-E62 E69-E70 E73-E74	E23-E25 E30-E31 E36-E40 E38-E42 E44-E48

CABLING FOR PARALLEL SYSTEM INTERFACING

The following paragraphs describe the interfacing required for parallel communication between iAPX 86, 88-based boards and the various Intellec systems.

Intellec Series II Model 210

The Intellec Series II Model 210 must be upgraded with disk drives in order to use it with the iSBC 957B package.

iAPX 86, 88-based boards interface to the UPP port of the Intellec Series II Model 210. Connect the cables as shown in Figure 2-4. Secure the parallel connector to the Intellec chassis using two 4-40 panhead screws.

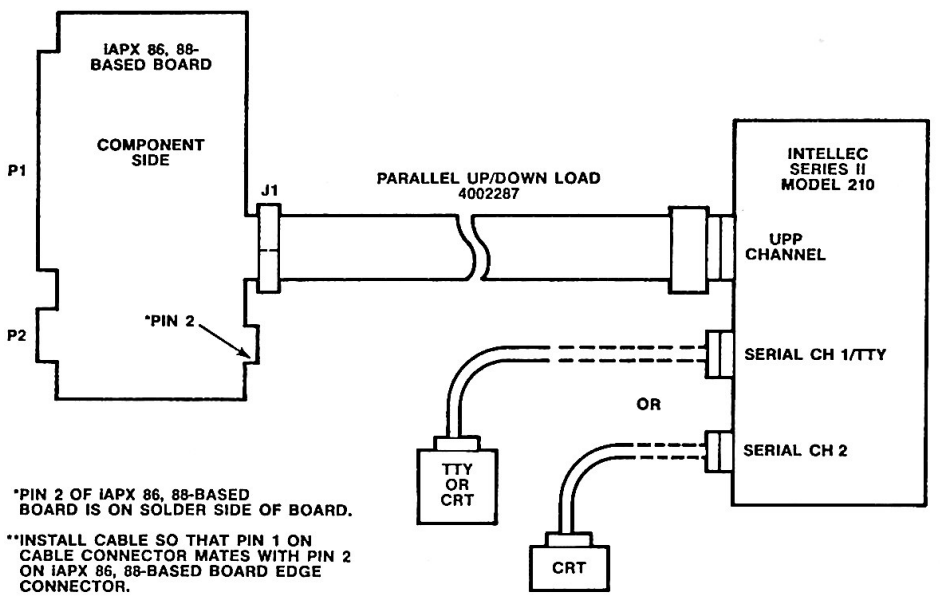


Figure 2-4. iAPX 88, 88-Based Board to Intellec Series II Model 210 Parallel Interface Cabling

Intellec Series II Model 220/230

iAPX 86, 88-based boards interface to UPP port of the Intellec Series II Model 220 or Model 230. Connect the cables as shown in Figure 2-5. Secure the parallel connector to the Intellec chassis using two 4-40 panhead screws.

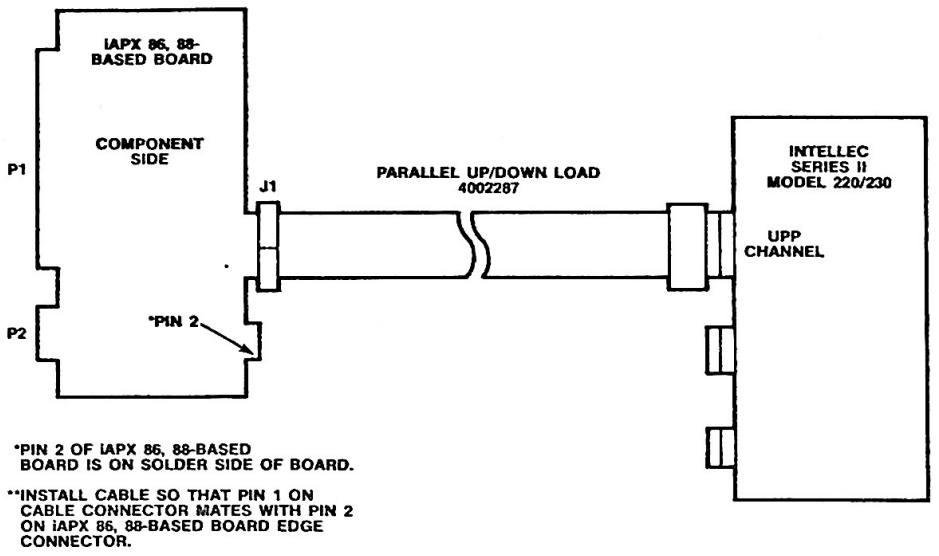


Figure 2-5. iAPX 86, 88-Based Board to Intellec Series II Model 220/230 Parallel Interface Cabling

Intellec 800

iAPX 86, 88-based boards can be interfaced to the UPP port of the Intellec 800. Connect the cables as shown in Figure 2-6. Secure the parallel connector to the Intellec chassis using two 4-40 panhead screws.

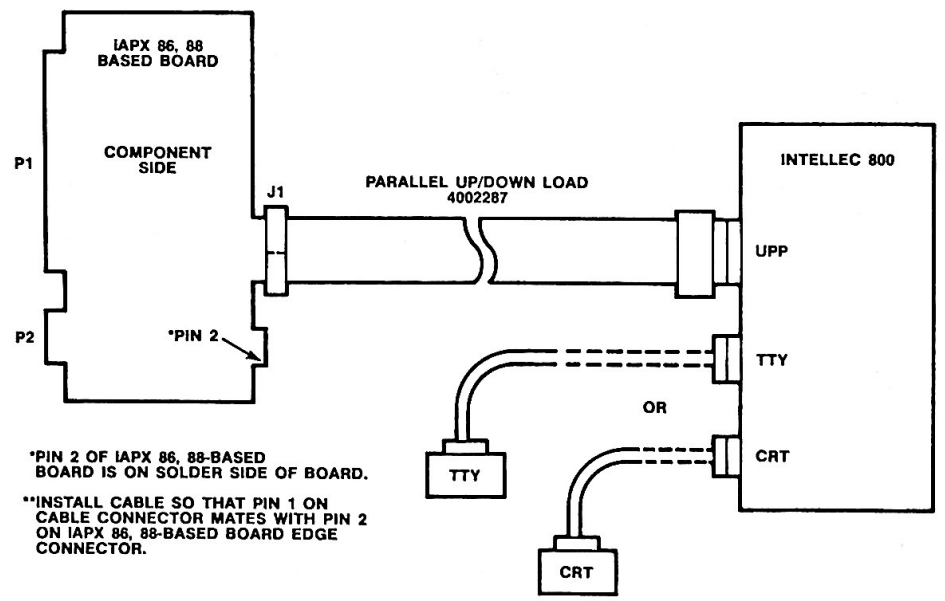


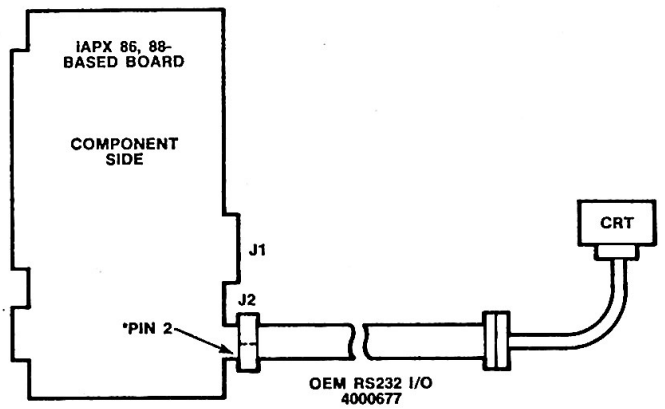
Figure 2-6. iAPX 86, 88-Based Board to Intellec 800 Parallel Interface Cabling

INTERFACING WITHOUT AN INTELLEC SYSTEM

If an Intellec System is not available, the iAPX 86, 88-based board may be connected directly to a CRT console device. Connect the jumpers on the board for serial operation and connect the cable to the serial port as shown in Figure 2-7. The supported baud rates for this type of interface are listed in the section on the BAUD_RATE macro in Chapter 6.

NOTE

The cable that connects from the CRT to the RS232C flat cable is not supplied with the iSBC 957B Intellec package.



*PIN 2 OF iAPX 86, 88-BASED BOARD IS ON SOLDER SIDE OF BOARD.

**INSTALL CABLE SO THAT PIN 1 ON CABLE CONNECTOR MATES WITH PIN 2 ON iAPX 86, 88-BASED BOARD EDGE CONNECTOR.

Fig. 2-7. iAPX 86, 88-Based Board to CRT Console Device

CHAPTER 3. OPERATION

This chapter provides operating instructions for entering iSBC 957B commands. Four of these commands (E, L, R, and T) are for the loader and require an Intellec Development System. The remainder of the commands are for the monitor and can be entered either at a terminal connected directly to an iAPX 86, 88-based board or on the console of an Intellec system which is connected to a iAPX 86, 88-based board. The 8087 Numeric Processor Extension is referred to as the "NPX" in the remainder of this manual.

START-UP PROCEDURE

To activate the monitor from an Intellec terminal, first make certain the correct cabling is connected between the Intellec system and the iAPX 86, 88-based board (processor board). Then power-on or reset the processor board. Invoke the loader program, APXLOD, by entering from the Intellec console the command:

```
-:Fn:APXLOD
```

which will cause the following display:

```
ISIS-II iAPX 86, 88 LOADER Vx.x

iAPX 86, 88 Monitor Vx.x
.
```

In the command, :Fn: specifies the number of the drive containing the diskette with the loader program and Vx.x denotes the current version of the monitor and loader. At this point in the display, the monitor's prompt, a period (.) is issued and the loader or monitor is ready to accept commands.

It is also possible to use the monitor without the Intellec system. The hardware for such an application includes a CRT connected directly to an iAPX 86, 88-based board through a serial port. To bring up the monitor in such a configuration, turn on the CRT and then reset or power-on the processor board. If the CRT is set up for 9600 Baud, the monitor will prompt by outputting a series of asterisks (*) to the CRT. If the CRT is not set up for 9600 Baud, other characters (possibly none) will be output. Regardless of which character is displayed on the CRT, next type an upper case "U" twice on the keyboard. This activates the monitor, which sends to the CRT the message

```
iAPX 86, 88 MONITOR, Vx.x
.
```

followed by the monitor's prompt. At this point, the monitor is ready to accept commands. See the section on the BAUD_RATE macro on Chapter 6 for a list of supported CRT baud rates.

NOTE

If the start-up procedure is unsuccessful, try again. If your hardware configuration includes an Intellec system, push interrupt button number 1 prior to retrying the start-up procedure. This will reestablish communication with the ISIS-II operating system.

COMMAND STRUCTURE

Responses to the monitor's command-level prompt are line-oriented, as opposed to the more traditional character-oriented monitor input. This allows for command-line editing capabilities.

Each monitor command includes a key letter, which is suggestive of the function of the command, such as D for displaying memory and S for substituting memory. Some commands have one or more additional letters which specify variations of the general function.

Following the key letter or letters of a command are zero or more arguments. The arguments can be addresses, data, register names, strings, or punctuation symbols dpending on the command.

In the remainder of this manual, the following syntax conventions are used:

[A]	indicates that "A" is optional
[A]*	indicates zero or more optional iterations of "A"
<B>	indicates that "B" is a variable
{A|B}	indicates "A" or "B"
<cr>	indicates a carriage return

Variables in commands include numbers, registers, expressions, and addresses. The BYTE and WORD variables are defined below.

BYTE AND WORD VARIABLES

```
<dec digit>::= {0|1|2|3|4|5|6|7|8|9}
<hex digit>::= {<dec digit>|A|B|C|D|E|F}
<dec number>::= {<dec digit><dec number>|<dec digit>}
<hex number>::= {<hex digit><hex number>|<hex digit>}
<number>::=  {<hex number>|<dec number>T}
<register>::=  {AX|BX|CX|DX|SP|BP|SI|DI|CS|DS|SS|ES|IP|FL}
```

```
<term>::= {<number>|<register>}
<expr>::= {<term>|<expr>{+|-}<term>}
<addr>::= {[<expr>:]<expr>}
<range>::= {<addr>|<addr>#<number>}
```

The range of byte values is 00-0FFH. Larger numbers may be entered but only the last two digits are significant because the number is evaluated modulo 256. The range of word values is 0000-0FFFFH. Larger numbers may be entered, but only the last four are significant because the number is evaluated modulo 65536. Leading zeros may be omitted for both types of values.

Byte and word values are assumed to be in hexadecimal. However, decimal values can be entered if they end with a "T". The trailing "H" that sometimes indicates hexadecimal is not allowed for byte or word values.

When word values are displayed, the contents of the high byte of the address location is displayed, followed by the contents of the low byte of the address location. Similarly, when entering word values, the high byte is followed by the low byte. If necessary, leading zeros will be appended to the value by the monitor. Assume, for example, that the byte values C4, 26, F2, and 3D are in consecutive locations beginning at 246B:26. A display of those locations in bytes looks like:

```
246B:0026 C4 26 F2 3D
```

while the corresponding display in words looks like:

```
246B:0026 26C4 3DF2
```

NUMERIC (REAL, INTEGER AND BCD) VARIABLES

```
<sign>::= [{+|-}]
<npx dec number>::=  <sign><dec number>
<npx hex number>::=  <hex number>H
<scientific number>::= {<npx dec number>[.<dec number>]|
                        <sign>.<dec number>}[E<npx dec number>]
<int number>::= {<npx dec number>|<npx hex number>}
<BCD number>::= {<npx dec number>|<npx hex number>}
<real number>::= {<scientific number>|<npx dec number>|<hex number>R}
<npx register>::= {CW|SW|TW|OP|DP}
<npx stack register>::=  ST[({0|1|2|3|4|5|6|7})]
```

Numeric variables refer to the data types supported by the 8087 Numeric Processor Extension (the NPX). There are three types of numeric variables: integer, packed binary coded decimal (BCD) and real. Of these three basic types, the integer and real types have three sub-types. All seven numeric data types are described in Table 3-1. For the remainder of this manual, the seven numeric variables will be referred to as "NPX data types".

Table 3-1. NPX Data Types

Data Type	Explicit Suffix	Bits	Significant Digits (Decimal)	Approximate Range (Decimal)
Word integer	H	16	4	$-32,768 < X < +32,767$
Short integer	H	32	10	$-2x10^9 < X < +2x10^9$
Long integer	H	64	19	$-9x10^{18} < X < +9x10^{18}$
Packed decimal	H	80	18	-99..99 < X < +99..99(18 digits)
Short real*	R	32	6-7	$8.43x10^{-37} < \|X\| < 3.37x10^{38}$
Long real*	R	64	15-16	$4.19x10^{-307} < \|X\| < 1.67x10^{308}$
Temporary real	R	80	19	$3.4x10^{-4929} < \|X\| < 1.2x10^{4929}$

* The short and long real data types correspond to the single and double precision data types defined in other Intel numeric products.

See the 8086 FAMILY USER'S MANUAL NUMERICS SUPPLEMENT for more details on the NPX data types. Also, note the section on "Constants" in the 8086/8087/8088 MACRO ASSEMBLY LANGUAGE REFERENCE MANUAL. For other NPX related details, refer to the Application Note, Getting Started With the Numeric Data Processor.

The suffixes used when entering the NPX data types differ from the suffixes for word and byte variables. If the no suffix is given when entering an NPX data type, the number is assumed to be a decimal number. A decimal number is defined for the real NPX data types as a value entered as a scientific number. This allows values like 4, 1.2, -1.2, -.3, -.3E-44, -1.56E-999 or 5.67E55 to be entered. A decimal number is defined for the integer and BCD NPX data types as a value entered as a scientific number that will evaluate to an integer value. This allows numbers like 12, -12, 4E2 or 4.0E1 to be entered but won't allow the entry of numbers like 1.2, -1.2 or -1.56E-999. In the valid cases, the monitor will place the hexadecimal equivalent of the input decimal number into iAPX 86, 88 memory. However, if an integer or BCD number is entered

with its explicit suffix "H" or a real number is entered with its explicit suffix "R", the monitor will place the number, as it is entered at the console, into iAPX 86, 88 memory. In this case explicit signs (+ or -) are not allowed, the hexadecimal number, entered at the console, indicates the sign of the number in the sign bit, the most significant bit.

When NPX data types are displayed, the address of the data type is displayed and then the value there is displayed in hexadecimal form. The number is then displayed as the equivalent decimal number if it has an equivalent decimal value. For example, the long real number 11223344, is displayed in form:

```
1111:0 4165682600000000R     11223344
```

The long integer, 11223344, is displayed in the form:

```
1111:0 0000000000AB4130H     11223344
```

The BCD number, 11223344, is displayed in the form:

```
1111:0 00000000000011223344T  11223344
```

In the remainder of this manual this display form is referred to as "NPX number format". If the memory value is a special bit pattern identifying non-numeric values like Not-A-Number (NAN) or Infinity, the address and the hexadecimal number are displayed and then the meaning is shown as NAN or Infinity instead of the decimal value. Examples of these displays using a long real number are:

```
0080:0000  FFFF000000000000R  -NAN
0080:0008  7FF0000000000000R  +Infinity
```

Special cases of numeric values are also identified. A negative zero is dislayed as -0. Pseudo zeroes (zero fraction with non-zero exponent) are shown as 0Eexp, where exp is the base 10 power equivalent of the binary exponent in the number. Numbers which are not normalized (I bit is zero) are displayed with their hexadecimal value and a "Bit" value which is a count of how many leading zeroes existed in the number. This "Bit" value indicates how many times the fractional part of the number must be shifted to the left to normalize it. An example of this display using a temporary real number is:

```
0080:000  3FFF199999999999999AR  .2 UNNORM 3 BITS
```

Decimal values may be displayed in any of four different formats. The format used depends on the range of the number and its value. Numbers which are exact integers and fit in the field size of 16 digits are displayed as integers with no trailing decimal point or 0. An example of this display using a long real number is:

```
0080:0000  43118B54F22AEB00R  1234567890123456
```

Values which appear as integers, within the limits of the field size, but are not exact integers are displayed as XXXXX.0. The .0 suffix indicates that the value is close to an integer but not exactly. An example of this display using a long real number is:

0080:0000 42DC12218377DE46R 123456789012345.0

If the magnitude of a number is greater than or equal to 0.1 and is less than 10**<field size>, the number will be displayed as XXXX.XXX. An example of this display using a long real number is:

0080:0000 41D26480B487E69BR 1234567890.12345

Finally, very large or very small numbers will be displayed in scientific number format X.XXXXXEexp. An example of this display using a long real number is:

0080:0000 492C2916217B84B7R 3.14E+44

Trailing zeroes after the decimal point will also be suppressed.

When NPX data types are displayed, the contents of the most significant byte of the numeric memory location is displayed in the leftmost position of the hexadecimal display, followed by bytes of decreasing significance and finally the least significant byte is displayed in the right most position of the hexadecimal display. Similarly, when entering NPX data types in hexadecimal or decimal, the first digit entered has the greatest significance and successive digits entered have decreasing significance. If less than the NPX data type's number of significant digits is entered, the monitor will append leading zeros. When entering a value for an NPX date type in scientific number format, the number is converted to its hexadecimal equivalent and is then stored in iAPX 86, 88 memory in that format.

ADDRESS SPECIFICATION

A complete address argument consists of a base and an offset separated by a colon (:). If the optional base portion is omitted, the contents of the iAPX 86, 88 CS register are used as a default base, except as noted in the command descriptions below. If an entire address is omitted, but an address is needed in the command, the contents of the CS and IP registers are used, respectively, as base and offset, except as noted in the command description.

There are two ways of denoting a range of addresses. One way is to list both the starting and ending addresses, with an exclamation point between them. An example is 30:46D ! 30:4FE. The other way is to list the starting address and the length in bytes, with a pound sign (#) between them. An example equivalent to the earlier one is 30:46D # 92.

If the ending address in a range lacks an explicit base part, the base of the starting address is assumed. The ending address may not contain a base part which differs from the base part of the starting address.

The largest count or the maximum number of bytes specified by a range is OFFFFh. When a range is expected and neither an ending address nor a length is specified, the range is taken to be a single byte.

MULTIPLE COMMANDS ON A SINGLE LINE

There are two mechanisms for putting more than one command on a command line. First, separate commands may be in the same command line if they are separated by semicolons (;). Second, by enclosing a command in angle brackets (<command>) and by placing a decimal repetition factor ahead of the first bracket, you can specify that the command be repeated the desired number of times. A repetition factor of n says "do this command n times." For example,

```
5 <12 <G, CS:3B7> ; D DS:4A>
```

is a valid command line that is built from the commands G, CS:3B7 and D DS:4A. The command G, CS:3B7 is repeated 12 times, then the D DS:4A is performed once. This entire sequence is repeated 5 times so the G, CS:3B7 command is repeated at total of 60 times while the D DS:4A command is repeated a total of only 5 times. Note that this use of angle brackets is NOT the same as the use of angle brackets in the syntax definition.

Closely related to repetition, but differing, is continuation. By putting a decimal continuation factor, n, immediately ahead of a command's key letter or letters, you are directing the monitor to "do this command for n items at a time." For example, the command D 200:14 directs the monitor to display the byte at address 200:14, while 20D 200:14 causes the display of 20 consecutive bytes, beginning at address 200:14. In contrast, 20 <D 200:14> causes the byte at 200:14 to be displayed 20 times.

NOTE

Both repetition and continuation factors are written as positive decimal integers with no "T" suffix. The range of these factors is 1-65,535. In any other part of a command using byte or word variables, however, decimal integers must have a "T" suffix, such as 127T.

iAPX 86 AND iAPX 88 CPU REGISTERS

The iAPX 86 and iAPX 88 CPUs include the 14 registers listed in Table 3-2. The abbreviations used in the table are those used in the command syntax.

Table 3-2. iAPX 86, 88 CPU Registers

Register Name	Abbreviation
Accumulator	AX
Base	BX
Count	CX
Data	DX
Stack Pointer	SP
Base Pointer	BP
Source Index	SI
Destination Index	DI
Code Segment	CS
Data Segment	DS
Stack Segment	SS
Extra Segment	ES
Instruction Pointer	IP
Flag	FL

NPX REGISTERS

The NPX includes the eight 80 bit individually addressable stack registers plus the status word, control word, tag word, instruction pointer, data pointer and instruction opcode field listed in Table 3-3. The abbreviations used in the table are those used in the command syntax. Note that the NPX instruction pointer listed below is not the iAPX 86, 88 instruction pointer. The monitor contains no command to modify the NPX instruction pointer.

Table 3-3. NPX Registers

Register Name	Abbreviation
NPX State	N
Status Word	SW
Control Word	CW
Tag Word	TW
Instruction Pointer	IP
Data Pointer	DP
Instruction Opcode	OP
Stack Register 0	St(0)
.	.
.	.
.	.
Stack Register 7	St(7)

ERRORS

Each line input to the monitor is checked for validity. If the command is invalid or impossible to execute, an explanatory error message is displayed. If the command line containing the error consists of multiple commands, any good commands prior to the error will be executed.

Three error messages - "Bad EMDS Connection", "Bad Patch Byte=hex number", and XISIS Abort" - are all indicative of hardware problems. To recover, check your hardware, restart monitor, and try again.

ENTERING COMMANDS

The monitor's command line editor responds to input as follows:

- Digits, upper and lower case letters, and all other standard keyboard characters are accepted into the command line and are printed on the console. Upper and lower case letters are indistinguishable to the monitor, all display is done by the monitor in upper case.

- RUBOUT deletes the most recently entered character (with backspace, space, backspace) from both the command line and the display. An attempt to rubout the prompt causes a beep to be sounded.

- CONTROL-C directs the monitor to abort its current command and issue a prompt. However, if your program is running and is in a loop, control-C does not have any effect.

- CONTROL-R displays at the console the current command line. If the consol is directly connected to an iAPX 86, 88-based board, however, control-R has no effect.

- CONTROL-X deletes the current command line and displays a pound sign (#).

- CONTROL-S causes the console output to be suspended at the current cursor position. No output is lost by this command.

- CONTROL-Q causes the console output, suspended by Control-S, to be resumed beginning at the current cursor position.

- CARRIAGE RETURN signals the completion of the command line, which is then read and acted upon.

- All other characters have no effect. Spaces may be included anywhere in the command line except within lexical elements.

- NPX data types may be entered only on the substitute ("S") or "XST(n)" command line and may appear in no other command line.

Command lines may be up to 255 characters in length. An attempt to exceed this limit will be unsuccessful and will cause the terminal to beep.

COMMAND DESCRIPTIONS

The following paragraphs describe the loader and monitor commands. Where appropriate, examples are provided. The commands are summarized in Table 3-4. In the examples of the commands all user input is underlined.

Table 3-4. Summary of Loader And Monitor Commands

COMMAND	FUNCTION AND SYNTAX
L Load	Loads an absolute object file from Intellec into iAPX 86, 88 memory. L <filename> <cr>
G Go	Transfers control of the CPU to the user program. G [<start-addr>][, <break-addr>|<range>]<cr>
R Load and Go	Loads an absolute object file from Intellec into iAPX 86, 88 memory and begins execution. R<filename><cr>
T Upload	Loads a block of iAPX 86, 88 memory into an Intellec file. T<range> , <filename> [, <start-addr>]<cr>
N Single Step	Displays and executes one instruction at a time. [<cont>] N [O] [P] [Q] [<start-addr>][,]<cr>
X Examine	Displays or modifies iAPX 86, 88 or NPX registers. X[<reg>[=<expr>]]<cr> X{N|[<npx register>[=<hex number>]]| [<npx stack register>[=<real number>]]}<cr>
D Display	Displays contents of a memory block. [<cont>] D [{W|I|SI|LI|T|SR|LR|TR|X}] [<range>][,]<cr>
S Substitute	Displays/modifies memory locations. [<cont>] S [W]<addr>[=<expr>][/<expr>]*[,]<cr> [<cont>] S [{I|SI|LI}] <addr>[=<int number>] [/<int number>]*],[<cr> [<cont>] S [{SR|LR|TR}]<addr>[=<real number>] [/<real number>]*[,]<cr> [<cont>] S [T]<addr>[=<BCD number>][/<BCD number>]*[,]<cr>
M Move	Moves the contents of a memory block. M<range> , <dest-addr><cr>
F Find	Searches a memory block for a constant. F<range> , <data><cr>
C Compare	Compares two memory blocks. C<range> , <dest-addr><cr>

Table 3-4. Summary of Loader And Monitor Commands (continued)

COMMAND	FUNCTION AND SYNTAX
I Input	Inputs and displays data from input port. [repeat] I [W]<port-addr><cr>
O Output	Outputs data to output port. [repeat] O [W]<port-addr> , <data> <cr>
P Print	Prints values or literals. P [{T|S|Q}][{<addr>|<expr>|<literal>}] [, {<addr>|<expr>|<literal>}]*<cr>
E Exit	Exits the loader program and returns to ISIS-II. E<cr>
* Comment	Rest of line is a comment. * <comment><cr>
B Bootstrap	Bootstraps code from iRMX 86 or 88 file compatible peripherals. B[<pathname>]

LOAD ABSOLUTE OBJECT FILE (L)

Function

This command (L) loads an absolute object file into the iAPX 86, 88 memory from the Intellec system.

Syntax

L<filename><cr>

Operation

The file indicated by <filename> must have been created by LOC86. The data read from the file is sent to the iAPX 86, 88-based board where it is loaded into the memory location specified in the file as established by the LOC86 command. If the file contains an execution starting address record, CS and IP are set to the indicated values.

The load command must be the only command in its command line.

In addition to the errors that the loader looks for, the load command is subject to ISIS-II I/O errors. It is an error if a file is loaded on top of monitor RAM. The locations of monitor memory are itemized in Chapter 5.

Example

.L:F1:PROG<cr>

The iAPX 86,88 absolute object file named PROG, which resides in drive 1, is loaded into iAPX 86, 88 memory. If PROG has a starting address, CS and IP are set to the appropriate values.

GO (G)

Function

This command (G) transfers control of the user program to the iAPX 86, 88-based board.

Syntax

G[<start-addr>] , [<break-addr>|<range>]*<cr>

Operation

Execution begins at the values in the CS and IP registers. If <start-addr> is included in the command line, IP and, optionally, CS are modified before execution begins.

In the command, up to four breakpoints can be specified. Two type of breakpoints may be set: execution breakpoints and memory breakpoints. An execution breakpoint occurs when the CPU attempts to execute an instruction at a <break-addr>. When an execution breakpoint occurs, the monitor prints

BREAK at <hex addr> : <hex addr>

and issues a prompt.

Each <range> specifies a memory breakpoint and can be up to 16 bytes in length. A memory breakpoint occurs when the CPU attempts to change the value at a location within the specified breakpoint range. When a memory breakpoint occurs, the monitor prints

BREAK with <hex addr> : <hex addr> = <hex number>

and issues a prompt.

NOTE

When setting an execution breakpoint at <start-addr>, and also when setting any memory breakpoint, be sure that there is room for six words to be pushed onto the stack. To ensure this, see that the stack, when allocated, has 12 extra bytes for this purpose.

For additional information relating to the Go command, see the section in Chapter 5 entitled "Special Considerations When Stepping."

Examples

```
.G<cr>
.G, 7FA, (DS:44#4)<cr>
.G, IP<cr>
```

In the first example, execution begins at the instruction whose address is given by CS:IP.

In the second example, execution begins at CS:IP and continues either until the instruction at CS:7FA is about to be executed or until at least one of the four bytes beginning at DS:44 is changed.

In the third example, execution begins at CS:IP and continues until the instruction at CS:IP is about to be executed again.

LOAD AND GO (R)

Function

This command (R) loads an iAPX 86,88 absolute object file with an execution starting address record from the Intellec system into iAPX 86, 88 memory. The loaded program then begins to execute at the specified starting address.

Syntax

```
R<filename><cr>
```

Operation

The file indicated by <filename> must have been created by LOC86. The data read from the file is sent to the iAPX 86, 88-based board where it is loaded into the memory location specified in the file as established by the LOC86 command. After loading, the CS and IP registers are set to the values indicated in the execution starting address record and execution begins at that location.

The load and go command must be the only command in its command line.

In addition to the errors that the loader looks for, the load and go command is subject to ISIS-II I/O errors. It is an error if a file is loaded on top of monitor RAM. The locations of monitor memory are itemized in Chapter 5.

Example

```
.R:F1:PROG<cr>
```

The iAPX 86, 88 absolute object file named PROG, which resides in drive 1, is loaded into iAPX 86, 88 memory. The starting address of PROG is loaded into the CS and IP registers, and execution commences there.

UPLOAD (T)

Function

This command (T) creates a new iAPX 86, 88 absolute object file on a disk in the Intellec system and then loads the contents of a block of iAPX 86, 88 memory into the new file. The name of the new file must be a valid ISIS-II file name.

Syntax

T<range>,<filename>[,<start-addr>]<cr>

Operation

The <range> defines the block of memory whose contents are to be unloaded into the new file whose name is specified by <filename>.

If the data in the <range> is a program, the starting address of the program can be specified in <start-addr>. If included, <start-addr> is placed in the proper place in the new file.

The upload command must be the only command in its command line.

In addition to the errors that the loader looks for, the upload command is subject to ISIS-II I/O errors.

Example

```
.T A64:4 ! A64:25C8, :F1:NEWFIL, A64:293<cr>
```

A program, residing in iAPX 86, 88 addresses A64:4 to A64:25C8, is uploaded to a new file called NEWFIL on a disk in drive 1 of the Intellec system. The starting address of the program is specified and is placed in NEWFIL.

SINGLE STEP (N)

Function

This command (N) causes a single instruction to be displayed in disassembled form.

Syntax

[<cont>]N[O][P][Q][<start-addr>][,]<cr>

Operation

If the optional continuation factor (cont) is present, that number of instructions will be displayed. All but the last of those instructions will also be executed.

If the O option is present, any routine called by an INT instruction will be treated as a single instruction. The INT instruction is displayed, the routine is executed, and then the next instruction after the routine is displayed.

The P option is just like the O option, except that it governs the treatment of routines that are called by a CALL instruction.

The Q option disables external processor interrupts while single stepping. The original state of the interrupt flag is restored when you are finished single stepping. However, if code executed by the single step command enables interrupts, the interrupt state will automatically be restored with interrupts enabled regardless of the original interrupt state. Use this option with caution.

The O, P and Q options may be used individually or in any combination that meets your needs and may be specified in any order.

Execution begins at the locations indicated by the values in the CS and IP registers. If <start-addr> is included in the command line, the specified IP and, optionally, a specified CS are used as the starting address for single step execution.

A comma signifies that the monitor will issue a special prompt symbol, a hyphen (-), after executing the single step instruction. Then the monitor waits for the user to enter another comma. When a comma is entered, the displayed instruction is executed, the next instruction is displayed, and the monitor issues another prompt for a comma. The cycle described in the previous sentence can be repeated as many times as you desire. When finished you can return to the monitor's general command level by entering a carriage return.

NOTE

Single stepping requires six words of user stack. To ensure that this is provided for, see that the stack, when allocated, has 12 extra bytes for this purpose.

For additional information regarding the Single Step command, see the section in Chapter 5 entitled "Special Considerations When Stepping".

Examples

1) .NOPQ<cr>

2) .24N 4, <cr>

In the first example, the current instruction is displayed and the monitor waits for input. If the next character entered is a comma, the displayed instruction is executed and the next instruction is displayed. If additional commas are entered, the process is repeated for each comma. All options are selected, calls and software interrupt routines will be treated as single instructions and interrupts are disabled until a carriage return is entered instead of a comma.

In the second example, execution starts at CS:4, 24 instructions are displayed, and the first 23 of those instructions are executed. If, subsequently, a comma is entered, 24 more instructions will be displayed and all but the last of them is executed.

EXAMINE/MODIFY REGISTERS (X)

Function

This command (X) is used to display and/or modify iAPX 86, iAPX 88 or NPX register contents.

iAPX 86 AND iAPX 88 REGISTERS

Syntax

X[<register>[=<expr>]]<cr>

Operation

If the command consists of only an X, all iAPX 86, 88 registers will be displayed and none of them is modified.

If only a register name is included in the command, the name of the register and its contents are displayed followed by a hyphen (-) prompt. If desired, a new hexadecimal value may be entered followed by a carriage return. This will change the register value. If no change is desired, just enter a carriage return.

If both the register name and an assignment expression are included in the command, the indicated assignment is immediate, no display of the new value occurs and there is no hyphen prompt.

Examples

1) .X<cr>

```
AX=FFFF SP=0428 CS=1000 IP=0000
BX=FFFF BP=FFFF DS=0000 FL=0000 = 00 D0 I0 T0 S0 Z0 A0 P0 C0
CX=FFFF SI=FFFF SS=0000
DX=FFFF DI=FFFF ES=0000
```

2) .X AX<cr>

```
AX = FFFF-
```

3) .X AX = BX + 2FD<cr>

(no display)

In the first example, all registers are displayed. The Flag register is also displayed by showing the binary value of the Overflow flag (O), Direction flag (D), Interrupt flag (I), Trap flag (T), Sign flag (S), Zero flag (Z), Auxillary flag (A), Parity flag (P) and Carry flag (C).

In the second example, the AX register value is displayed and a prompt is issued for either a new hexadecimal or decimal value in BYTE/WORD format or a carriage return, depending on whether or not you wish to change the register value.

In the third example, the value 2FDH is added to the value in BX and the result is placed in AX. No display of AX or BX occurs.

NPX REGISTERS AND STACK REGISTERS

Syntax

```
NPX State:
    XN<cr>
NPX Registers:
    X[<npx register>[=<hex number>]]<cr>
NPX Stack Registers:
    X[<npx stack register>[=<real number>]]<cr>
```

Operation

If the command consists of only an XN, the state of the NPX registers and stack registers is displayed and none of them is modified.

If only an NPX register name is included in the command, the name of the NPX register and its contents are displayed followed by a hyphen (-) prompt. If desired, a new hexadecimal value may be entered followed by a carriage return. This changes the value of the NPX register. If no change is desired, just enter a carriage return.

If only an NPX stack register name is included in the command, the name of the NPX stack register and its contents are displayed in NPX number format followed by a hypen (-) prompt. If desired, a value in real number format, as defined in Chapter 3, may be entered followed by a carriage return. This will change the value of the NPX stack register. If no change is desired, just enter carriage return.

If both the register name and an assignment expression are included in the command, the indicated assignment is immediate, no display of the new value occurs and there is no hyphen prompt.

All changes take effect immediately. This means that upon subsequent redisplay of the NPX state, any side effects will be shown. For example, not all bits of the 8087 control word can be set. If the control word is set to all ones then subsequently redisplayed, the value shown will have some control bits set to zeros. Also if the TOP bit of the status word is changed, the registers in the stack will rotate but their tag fields will not rotate to match the registers.

Examples

1) .XN <cr>

```
CW: X  X  X IC  R  C  P  C IM  X PM UM OM ZM DM IM
    0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  1  IP = 00000
SW: B C3  T  0  P C2 C1 C0 IR  X PE UE OE ZE DE IE
    0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  OP = 0000
TW: T  7  T  6  T  5  T  4  T  3  T  2  T  1  T  0
    0  0  0  0  1  1  1  0  1  0  0  0  0  0  0  1  DP = 00000
ST(0) ZERO     00000000000000000000R   0
ST(1) VALID    3FFF9999999999999999AR   1.2
ST(2) VALID    BFFF9999999999999999AR  -1.2
ST(3) SPECIAL  FFFF0000000000000000R  -Infinity
ST(4) SPECIAL  7FFFFF00000000000000R  +NAN
ST(5) EMPTY    4000C90FDA9E46A7843ER   3.14159265
ST(6) VALID    4CF5F08B8D41AF800AC8R   1.23456E+999
ST(7) VALID    3FFF1800000000000000R   .1875 UNNORM 3 BITS
```

2) .XCW<cr>

```
CW: X  X  X IC  R  C  P  C IM  X PM UM OM ZM DM IM
    0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  -
```

3) .XST(4)<cr>

```
ST(4) ZERO  00000000000000000000R   0
```

4) .XST = 1.23 E-4

In the first example all the NPX registers and stack registers are displayed. The CW, SW and TW registers are displayed in binary with a letter above the binary value indicating the meaning of the binary value. The data registers of the NPX are displayed in TOP relative form ST(0)-ST(7). The associated tag field is decoded and displayed as EMPTY, VALID, ZERO, or SPECIAL. The tag field is decoded independent of the data register's content. It is therefore possible to set the tag field to a value which does not correspond to the data register's content. The data value of all NPX stack registers is displayed in hexadecimal and decimal (if possible) independent of the register's tag. The DP (data pointer) and IP (NPX instruction pointer) fields are displayed as 5 hex digits. The OP (opcode) is shown as a 16 bit value, whose upper 5 bits will always be zero. See the 8086 Family User's Guide Numeric Supplement for a description of the CW (control word), SW (status word) and TW (tag word) registers.

In the second example, the CW register value is displayed and a prompt is issued for the entry of a hexidecimal value or a carriage return, depending on whether or not you wish to change the NPX register value.

In the third example, NPX stack register 4 is displayed and a prompt is issued for a real number, as defined in Chapter 3, or a carriage return, depending on whether or not you wish to change the register value.

In the fourth example, the real number 1.23E-4 is placed in NPX stack register 0 as the hexadecimal equivalent. No display of ST(0) occurs. Note that if no NPX stack register is specified, stack register zero is used.

DISPLAY MEMORY (D)

Function

This command (D) displays the contents of a specified block of memory; the contents can be displayed on a byte, word, word integer, short integer, long integer, BCD, short real, long real, temporary real or instruction basis.

Syntax

[<cont>]D[{W|I|SI|LI|T|SR|LR|TR|X}][<range>][,]<cr>

Operation

The optional continuation factor (cont) indicates the number of bytes, words, NPX data types or instructions to be displayed. It is ignored if the indicated range is more than one byte in length.

The association between the optional characters and the data types they designate is as follows:

Optional Character	Display Format
W	word
I	word integer
SI	short integer
LI	long integer
T	BCD
SR	short real
LR	long real
TR	temporary real
X	disassembled instruction

The absence of any of these options indicates that the display is to be both bytes and ASCII. If the byte value is not a valid ASCII character, a period (.) will be printed as its ASCII character.

If the optional <range> is omitted, the requested display begins with the byte, word, NPX data type or instruction at CS:IP.

A comma signifies that the monitor will issue a special prompt symbol, a hyphen (-), after executing the display instruction. Then the monitor waits for you to enter another comma. When a comma is entered, the next block of memory (equal in size to the one in the original request) is displayed, and the monitor issues another prompt for a comma. The cycle described in the previous sentence can be repeated as many times as desired. When finished, you can return to the monitor's general command level by entering a carriage return.

Examples

1) .14DX<cr>

```
0F00:012A 55       PUSH   BP
0F00:012B 8BEC     MOV    BP, SP
0F00:012D 8BF5     MOV    SI, BP
0F00:012F 83C608   ADD    SI, 8
0F00:0132 CDB8     INT    0B8H                    ; IMM = +184
0F00:0134 85C9     TEST   CX,CX
0F00:0136 7403     JZ     A = 013BH               ; $+5
0F00:0138 E82700   CALL   A = 0162H               ; $+42
0F00:013B C45E04   LES    BX,DWORD PTR [BP+04H]
0F00:013E 26890F   MOV    ES:[BX],CX
0F00:0141 5D       POP    BP
0F00:0142 C20600   RET    6                       ; NEAR
0F00:0145 C8       ??
```

2) .D DS:5#16T,<cr>

```
03E0:0005 0C 00 89 CE 88 54 2B 8B 46 47 31   *.....T+.FG1*
03E0:0010 D2 F7 F3 89 46                     *....F*,
03E0:0015 61 62 63 75 39 89 0E 0E 33 35 37   *abcu9...357*
03E0:0020 21 00 04 77 1E                     *!..w.*<cr>
```

3) .DW SS:SP ! SP+5<cr>

```
0470:0500 C28B 00CB 0000
```

4) .5DI <cr>

```
1000:0000   1ABDH    6845              0929H    2345
1000:0004   FFFFH   -1                 FBB6H   -1098
1000:0008   115CH    4444
```

5) .DT 1C0:0,

```
01C0:0000   0000000019283746543T      19283746543 -,
01C0:000A   0010203040506070809OT     1.0203040506070 81E+17 -,
01C0:0014   8000000000000000001T     -1 -<cr>
```

Examples (continued)

6) .5DTR 10<cr>

```
1000:0010   3FFF9D70A3D70A3D70A4R          1.23
1000:001A   4000C90F6F08CC575C07R          3.141567
1000:0024   3988CD06B506BD1138C3R          9.999E-499
1000:002E   00000000000000000000R          0
1000:0038   C04BF2405D408C3ADB0BR         -1.43E+23
```

7) .DX<cr>

```
0F00:149    D93E4E04   FSTCW     WORD PTR 044EH
```

8) .DX<cr>

```
0F00:149    D93E4E04   ESC       15, WORD PTR 044EH
```

In the first example, 13 instructions, starting at F00:12A, are displayed in disassembled form. Each line of the display consists of the starting address of the instruction (which is also the address of the op code), the instruction displayed in hexadecimal and the disassembled instruction. In the fifth line of the display, ";IMM = +184" means that the instruction contained an immediate reference to the decimal value +184. In the seventh line of the display, "A = 013BH" means that the JZ instruction is a short jump to CS:13B (F00:13B) if the zero flag is 1. The "; $+5 in this line indicates the destination address of the JZ op code is located 5 decimal bytes past the address given as the start address of the instruction. This convention is used for both short call and loop instructions. If the displayed instruction is a long call or a long jump, both the CS and IP values of the destination address are displayed. In the ninth line of the display, the type of variable referenced (in this case a DWORD or 4 byte value) is displayed. If an explicit segment override instruction (say to the code segment) were specified, the display of the type of variable reference would appear as "CS:DWORD PTR[BP + 04H]". In the twelfth line of the display, the "; NEAR" indicates that the return is a near return. The "??" in the thirteenth line means that the op code (in this case C8) is not an iAPX 86, 88 assembler instruction and therefore cannot be disassembled.

NOTE

When displaying the mnemonic and operands for an instruction or when single stepping, the monitor's disassembler always gives the number an explicit "H" suffix if the number is a hexadecimal value. If no suffix is displayed, the number is always a decimal value.

In the second example, 16 decimal bytes, beginning at DS:5, are displayed in both hexadecimal and ASCII (between the asterisks). After the display, a hyphen prompt is issued and the monitor waits for a comma to be typed. When a comma is entered, the next 16 bytes are displayed. If additional commas had been entered, the process would have been repeated for each comma. Note that each 16 byte display is split between two lines. This is because separate 16 byte paragraphs, or portions thereof, are displayed on separate lines starting at the beginning of the paragraph.

In the third example, the top three words on the stack are displayed.

In the fourth example, 5 word integers, beginning at 1000:0 (CS:IP used), are displayed in both hexadecimal and integer format.

In the fifth example, one BCD value, beginning at 1C0:0 is displayed in both BCD and decimal formats. A prompt is output to allow display of successive BCD values. The operator enters two commas, causing the display of a total of 3 BCD values.

In the sixth example, 5 temporary real values, beginning at CS:10, are displayed in both temporary real hexadecimal and decimal format.

In the seventh example, the NPX instruction "FSTCW" is shown as it is disassembled if the iAPX 86, 88-based board contains an NPX.

In the eighth example, the same NPX instruction "FSTCW" is shown as it is disassembled if the iAPX 86, 88-based board contains no NPX.

SUBSTITUTE

Function

The command (S) allows you to display and, optionally, to modify memory locations on a byte, word, or NPX data type basis.

Syntax

```
[<cont>] S [W]<addr>[=<expr>][/<expr>]*[,]<cr>
[<cont>] S [{I|SI|LI}]<addr>[=<int number>][/<int number>]*[,]<cr>
[<cont>] S [{SR|LR|TR}]<addr>[=<real number>][/<real number>]*[,]<cr>
[<cont>[ S [T]<addr>[=<BCD number>][/<BCD number>]*[,]<cr>
```

Operation

This command displays one memory location and then issues a hyphen (-) prompt. The prompt invites you to enter new values into successive locations beginning at the location displayed. If no changes are desired, entering a carriage return will terminate the command.

The optional continuation factor (cont) indicates the number of bytes, or words or NPX data types to be modified each time a continuation comma is requested.

The association between the optional character and the data types they designate is as follows:

Optional Character	Display Format
W	word
I	word integer
SI	short integer
LI	long integer
T	BCD
SR	short real
LR	long real
TR	temporary real

The absence of any of these options indicates that the substitution is to be on a byte basis.

The display begins at the location indicated by the values in the CS and IP registers. If <addr> is included in the command line, the address is used and CS and IP are not affected.

The presence of one or more expressions, separated by slashes, causes the indicated values to be placed in memory, beginning at the first location displayed.

If a comma follows the expression, then after that expression has been placed in memory, the next location following the last one modified is displayed and a hyphen (-) prompt is issued. In response to the prompt, you may enter a single expression, a list of expressions separated by slashes or you may terminate the command by entering a carriage return. If a list is entered, those values are placed in memory and a new prompt is issued. The process continues until a carriage return is entered.

Examples

1) .S<cr>

```
0070:0004  40 - 1/2/3,
0070:0007  20 - 4/15/<cr>
```

2) .100SW DS:AX=FFFF

3) .4SW DS:44<cr>

```
008A:0044 0001 - 9090,
008A:004C 0005 - 9090<cr>
```

Examples (continued)

4) .4SLR = -1.4E-66<cr>

5) .STR ES:AX = 1.2/3.141567/-1.2/-14.67E-67<cr>

In the first example, the current location is displayed, the prompt is issued, and the entered values 1, 2, and 3 are placed in locations 70:4 through 70:6. The comma says continue, so the monitor displays next value following the one last modified, issues another prompt, and places the entered values 4 and 15 in locations 70:7 and 70:8. The process is terminated by a carriage return.

In the second example, there is no display and FFFF is placed in 100 consecutive words of memory, beginning at location DS:AX.

In the third example, one word is displayed and 9090 is placed in four word locations starting at 8A:44. The comma says continue, so the next word following the one last modified is displayed and 9090 is placed in four word locations starting at 8A:48 modifying a total of 8 words of memory. The carriage return terminates the process.

In the fourth example, four long real values, starting at CS:IP, are changed to -1.4E-66.

In the fifth example, 4 temporary real values, starting at ES:AX, are set to 1.2, 3.141567, -1.2 and -1.467E-66 respectively. Note that the monitor will place the last temporary real value into standard scientific notation where there is only one digit to the left of the decimal point.

MOVE (M)

Function

This command (M) copies the contents of a block of memory.

Syntax

M<range>,<dest-addr><cr>

Operation

The block of memory defined by <range> is copied to the locations beginning at <dest-addr>.

Example

.M CS:2CD # 15T, 200:4A <cr>

This command line moves the 15 bytes beginning at CS:2CD to the location beginning at 200:4A.

FIND (F)

Function

This command (F) searches a block of memory for a sequence of hexadecimal digits.

Syntax

F<range>,<data><cr>

Operation

The block of memory indicated by <range> is searched through for the sequence of digits in the data argument. Each time a match is found, the address of the first matching byte is displayed.

The data sequence consists of from one to 32 hexadecimal digits, representing from one to 16 bytes. However, odd lengths from three to 31 are not allowed. If there is only one digit in the data sequence, it is treated as if it had been entered with a leading zero.

Example

.F CS:0 ! IP, 54455354<cr>

0200:0118
0200:01A4
0200:0212

In this example, the ASCII string "TEST" is found in three locations between CS:0 and CS:IP. In each, the first address of the string is displayed.

COMPARE (C)

Function

This command (C) compares the contents of one block of memory with that of another block.

Syntax

C<range>,<dest-addr><cr>

Operation

The block of memory defined by <range> is compared to the block of equal length that starts at <dest-addr>. For each mismatch encountered, both mismatching bytes are displayed in the format.

addr-one-byte one-byte other-byte addr-other-byte.

Example

User input is underscored.

.C CS:118 # 10, CS:1A4-5 <cr>

```
0200:0118 6E 28 0200:019F
0200:011A 67 4E 0200:01A1
0200:0123 3C 2D 0200:01AA
```

In this example, two memory blocks of length 16 bytes are compared. Three mismatches are found.

PORT INPUT (I)

Function

This command (I) inputs and displays a byte or word from the specified port.

Syntax

[<repeat>]I[W]<port-addr><cr>

Operation

The optional repeat factor indicates the number of bytes or words that are to be input and displayed. If the repeat factor is not present, it defaults to one.

The optional W indicates that words, not bytes, are to be input and displayed. If the W is not present, bytes will be input and displayed.

<port-addr> is the address of the port from which the data is to be obtained. <port-addr> may not have a base part.

Examples

```
.5I 2FA<cr>

FF
FF
FF
FF
FF

.IW 189<cr>

C923
```

In the first example, five bytes are obtained from port 2FA and are displayed. (The unchanging value probably indicates that there is no device attached to that port.)

In the second example, one word is obtained from port 189 and is displayed.

PORT OUTPUT (O)

Function

This command (O) outputs a byte or a word to the specified port.

Syntax

[<repeat>]O[W]<port-addr>,<expr><cr>

Operation

The repeat factor indicates how many times the specified byte or word is to be output to the specified port. If there is not a repeat factor, it is taken to be one.

The optional W indicates that the output mode is to be word. The absence of W indicates byte mode.

<port-addr> is the address of the port to which bytes or words are to be output. <port-addr> may not have a base part.

<expr> is an expression whose value is to be output.

Examples

1) .OW 3, AX + 1B9<cr>

2) .100 O 2CDE, 1<cr>

In the first example, the value of AX + 1B9 is output as a word to port number 3.

In the second example, 1 is output one hundred times as a byte value to port number 2CDE.

PRINT (P)

Function

This command (P) displays the value of an expression, the base and offset portions of an address, or a literal string of characters.

Syntax

```
P[{T|S|Q}][{<addr>|<expr>|<literal>}] [,
          {<addr>|<expr>|<literal>}]*<cr>
```

Operation

The optional T indicates that the values of expressions are to be printed in decimal form.

The optional S indicates that the values of expressionss are to be printed in signed decimal form.

The optional Q indicates that both quoted and unquoted strings are to be displayed as they appear in the command.

<addr> must be a complete address, with both base and offset portions. If only an offset is present, it is interpreted as an expression. When printed, the address has the form base:offset, where both base and offset are in hexadecimal.

A literal is a string of characters enclosed within single quotes.

In addition to acting as a separator between addresses, expressions, and literals, each comma causes a space to print.

To print a literal string that contains single quotes, each single quote must be entered twice.

Examples

User input is underscored.

1) .P DS:BX + SI + 7D<cr>

   00FB:145C

2) .P'AX + BX =',AX + BX<cr>

   AX + BX = 007D

3) .PS 72 - 119T<cr>

   -5

4) .P 'He said "Hello".'<cr>

   HE SAID 'HELLO'.

5) .PQ He said 'Hello'. <cr>

   HE SAID 'HELLO'.

In the first example, the values of DS and BX + SI + 7D are computed and displayed as the base and offset portions of an address.

In the second example, the literal 'AX + BX =' is printed, followed by the value of the expression AX + BX. Note that the comma in the command line caused a space to print immediately after the equal sign.

In the third example, the value of the expression 72H-119 is evaluated and printed as a signed decimal number.

In the fourth example, the single quotes with the literal string are made to print by representing each of them twice.

In the fifth example, the string is displayed exactly as it appears in the command.

EXIT (E)

Function

This command (E) transfers control of the Intellec system from the loader program to ISIS-II.

Syntax

E<cr>

Operation

The exit command must be on a command line by itself and it has no effect when entered into a terminal connected directly to an iAPX 86, 88-based board.

The state of the CPU on the iAPX 86, 88-based board when the monitor is reinvoked after exiting depends on whether the iAPX 86, 88-based board was reset in the interim. If resetting has been done, the monitor prints its usual sign-on message and the monitor environment is re-initialized. Otherwise, the monitor prints

Control-C

and the state of the iAPX 86, 88-based board is what it was when the exit command was executed. An exception to the latter case is when the 8080 in the Intellec system has done port I/O since the exit; in this event, it is best to reset the iAPX 86, 88-based board.

COMMENT (*)

Function

This command (*) causes the rest of the command line to be treated as a comment.

Syntax

*<comment><cr>

BOOTSTRAP (B)

Function

This command (B) loads the iRMX 86 Operating System, the iRMX 88 Real-Time Multitasking Executive or application code from iRMX 86 or 88 file compatible (target system) peripherals. It is a valid command only when you select the Bootstrap Loader option when configuring the monitor. See Chapter 6 for details on Bootstrap Loader configuration.

Syntax

B [<pathname>]<cr>

Operation

The file indicated by <pathname> must have been created by LOC86 or LIB86 and must be a valid iRMX 86 or 88 pathname. The data read from the file is sent to the iAPX 86, 88-based board where it is loaded into the memory address specified in the file as established by the LOC86 command. Execution automatically switches from the monitor to the bootstrapped code when the loading is completed.

If you do not specify <pathname> in the command line, the default file /SYSTEM/RMX86 will be loaded from iRMX 86 peripherals and /SYSTEM/RMX88 will be loaded from iRMX 88 peripherals.

If either the default file or the specified file do not exist on the bootstrap device(s), the Bootstrap Loader will halt and the monitor must then be reset to continue.

Examples

1) B<cr>

2) B :F1:MEMTST<cr>

In the first example, the default file (defined above) is loaded from the first ready peripheral defined in the Bootstrap Loader Device Table.

In the second example, the file, MEMTST, is loaded from the device configured as :F1:.

iRMX 86 users should refer to the iRMX 86 Loader Reference Manual and iRMX 88 users should refer to the iRMX 88 Reference Manual for details on the functioning of the Bootstrap Loader.

CHAPTER 4. SYSTEM I/O ROUTINES

This chapter presents several ISIS-like routines that can be called by user programs running on an iAPX 86, 88-based board. All of the routines in this chapter, except CI (console input), CO (console output) and a special case of READ, require an Intellec Development System. CI and CO can be used with a terminal connected directly to an iAPX 86, 88-based board. Table 4-1 summarizes, for convenient reference, the calling sequences for the system I/O routines in this chapter. In the table, the comments after some of the calls contain, in order, abbreviated data types for the call's arguments; "p" stands for POINTER, "w" stands for WORD, and "b" stands for BYTE.

Table 4-1. Summary of System I/O Routines

```
CALL OPEN(AFTN, FILE, ACCESS, ECHOAFTN, STATUS); /*p,p,w,w,p*/
CALL READ(AFTN, BUFFER, COUNT, ACTUAL, STATUS);  /*w,p,w,p,p*/
CALL WRITE(AFTN, BUFFER, COUNT, STATUS);  /*w,p,w,p*/
CALL SEEK(AFTN, MODE, BLOCKNO, BYTENO, STATUS);  /*w,w,p,p,p*/
CALL RESCAN(AFTN, STATUS);  /*w,p*/
CALL CLOSE(AFTN, STATUS);  /*w,p*/
CALL DELETE(FILENAME, STATUS);  /*p,p*/
CALL RENAME(OLDNAME, NEWNAME, STATUS);  /*p,p,p*/
CALL ATTRIB(FILENAME, ATTRIBUTE, ONOFF, STATUS);  /*p,w,w,p*/
CALL LOAD(FILENAME, BIAS, SWITCH, ENTRY, STATUS);  /*p,w,w,p,p*/
CALL ERROR(ERRNUM);  /*w*/
CHAR = CI;
CALL CO(CHAR);  /*b*/
CALL ISIS(FUNCTION, PARAM$BLOCK);  /*w,w*/
CALL EXIT;
```

LIBRARIES FOR PL/M-86 CASES

The diskette on which the loader resides also includes three libraries. The libraries contain ISIS-like routines that can be called by user programs running on the iAPX 86, 88-based board. The routines in all three libraries have the same names and functions, but three libraries are necessary to support the three types of subroutine linkages provided by the different models of computation provided by the PL/M-86 compiler. See the PL/M-86 USER'S GUIDE FOR 8086-BASED DEVELOPMENT SYSTEMS for information about the cases.

In the following sections, the calling sequences for the routines are described in detail. Bear in mind that the descriptions apply to all four PL/M-86 cases, "small", "compact", "medium", and "large". In the case of the "small" control, however, the following also apply:

The routines in APXIOS.LIB (for the "small" case) and APXIOC.LIB (for the "compact" case) are written to be called with intrasegment subroutine calls. The routines in APXIOL.LIB (for the "medium" and "large" cases) are written to be called with intersegment subroutine calls.

Small & compact - intra segment subroutine calls
medium & large - inter segment subroutine calls

NOTE

Programs compiled with the "small" control and the ROM option should be linked to routines in APXIOC.LIB, not APXIOS.LIB.

The routines in all three libraries were written in PL/M-86. The modules in APXIOS.LIB were compiled with the "small" control, those in APXIOC.LIB were compiled with the "compact" control, and those in APXIOL.LIB were compiled with the "large" control. The names assigned to the segments, classes, and groups are the standard names generated by the PL/M-86 compiler.

- Arguments described as having to be POINTERS may also be WORDS or ADDRESSES in the "small" case. Similarly, arguments described as having to be WORDS may also be POINTERS or ADDRESSES. This is because, in the "small" case, the three terms refer equivalently to two byte quantities. However, if compilation is done with the ROM option, POINTERS are four byte quantities, while WORDS and ADDRESSES are still two byte quantities.

- In the examples that follow, the commercial "at" sysmbol (@) is used to indicate the address of a variable name rather than its contents. In the "small" case, a period (.) may be used for this purpose.

OPEN ROUTINE

A call to the OPEN routine initializes ISIS-II tables in the Intellec system and allocates buffers required for I/O processing of the specified file.

The arguments that must be passed in a call to OPEN are the following:

AFTNPTR — A POINTER to a WORD in which the monitor stores the active file number (AFTN) of the file that is opened. The AFTN is used to refer to the file in other calls to the routines of this chapter. :CI: and :CO: are always open and have the AFTNs 1 and 0 permanently assigned. Excluding :CI: and :CO:, only six files can be open at a time.

FILE A POINTER to an ASCII string containing the name of the file to be opened. The ASCII string can contain leading space characters but no embedded space characters. It must be terminated by a character other than a letter, digit, colon (:), or period (.). A space may be used as a terminating character.

ACCESS A WORD containing the access mode for which the file is being opened. A value of 1 specifies that the file is open only for input to the system (READ). A value of 2 specifies that the file is open only for output from the system (WRITE). A value of 3 specifies that the file is open for updating (READ or WRITE). When a file is open for input, MARKER is set to 0 and LENGTH is unchanged. If the file specified for input is nonexistent, a nonfatal ISIS error occurs. When a file is opened for output, MARKER and LENGTH are set to 0. If the file specified for output is nonexistent, a disk file is created with the specified filename and all attributes of the new file are reset. Specifying a disk file whose format or write-protect attributes are set causes a nonfatal ISIS error. When a file is opened for updating, MARKER is set to 0. If the file already created with the specified filename and all attributes reset. LENGTH is set to 0. Opening a file for an access mode that is not physically possible causes a nonfatal ISIS error.

ECHOAFTN A WORD containing the AFTN of the echo file if the file is to be opened for line editing. The echo file must previously have been opened. The AFTN of the echo file is passed in the less significant byte of the field. If this field contains 0, no line editing is done. To specify an AFTN of 0 for :CO:, a nonzero value must be in the more significant byte and 0 in the less significant byte. For example, F000H specifies the AFTN for the :CO: device.

STATUS A POINTER to a WORD to which the monitor will return a status code in the event of a nonfatal ISIS error. The errors and their numbers are listed in Appendix A.

Example

The following PL/M-86 program segments illustrate using the OPEN command to open a file called INFILE for transferring data from INFILE to the console. The file resides in drive 1. Note the file name is terminated by a space.

```
OPEN:
    PROCEDURE (AFTN, FILE, ACCESS, ECHOAFTN, STATUS) EXTERNAL;
      DECLARE (AFTN, FILE, STATUS) POINTER;
      DECLARE (ACCESS, ECHOAFTN) WORD;
    END OPEN;
```

Example (continued)

```
      .
      .
      .
DECLARE NEW$AFTN WORD;
DECLARE STATUS$VALUE WORD;
      .
      .
      .
CALL OPEN(@NEW$AFTN, @(':F1:INFILE '), 1, 0, @STATUS$VALUE);
```

READ ROUTINE

A call to the READ routine transfers up to 4096 bytes of data from an open file to a memory location specified by the calling program. See the section entitled "Line-Edited Input Files" in the ISIS-II USER'S GUIDE for information concerning inputting line-edited files. When AFTN is 1, the READ routine may be used without an Intellec system to provide line edited input.

The arguments that must be passed in a call to READ are the following:

AFTN — A WORD containing the active file number of a file that is open for reading or updating. The AFTN value either was returned by a previous call to the OPEN routine or it is 1 for :CI:.

BUFFER — A POINTER to a buffer to which data is to be transferred from the open file. The length of the buffer must be at least as great as the value of the COUNT argument. If the buffer is too short, the memory locations that follow the buffer will be overwritten.

COUNT — A WORD containing the number of bytes to be transferred from the file to the buffer.

ACTUAL — A POINTER to a WORD to which the number of bytes actually transferred is to be sent. The same number is added to MARKER. The number of bytes transferred is never more than COUNT. For line-edited files, the number of bytes transferred is never more than the number of bytes in the line-editing buffer. When a file is not line-edited, the number of bytes transferred is equal either to COUNT or to LENGTH minus MARKER, whichever is smaller. An end-of-file condition is indicated when the number of bytes transferred is less than COUNT (except for line-edited files) or when the number of bytes transferred equals zero (unless COUNT equals 0), in which case no bytes are transferred.

STATUS — A POINTER to a WORD to which the monitor will return a status code in the event of a nonfatal ISIS error. The errors and their numbers are listed in Appendix A.

Example

The following PL/M-86 program segments illustrate using the READ routine to transfer 128 bytes from a file whose AFTN is in FILE$AFTN to a buffer in memory.

```
READ:
    PROCEDURE (AFTN, BUFFER, COUNT, ACTUAL, STATUS) EXTERNAL;
      DECLARE (AFTN, COUNT) WORD;
      DECLARE (BUFFER, ACTUAL, STATUS) POINTER;
    END READ;
      .
      .
      .
    DECLARE FILE$AFTN WORD;
    DECLARE BUFFER(128) BYTE;
    DECLARE TRANSFERRED WORD;
    DECLARE STATUS$VALUE WORD;
      .
      .
      .
    CALL READ(FILE$AFTN, @BUFFER, 128, @TRANSFERRED, @STATUS$VALUE);
      .
      .
      .
```

WRITE ROUTINE

A call to the WRITE routine transfers up to 4096 bytes of data from a specified location in memory to a file open for writing or updating.

The arguments that must be passed in a call to WRITE are the following:

AFTN — A WORD containing the active file number of a file that is open for writing or updating. The AFTN value either was returned by a previous call to the OPEN routine or it is 0 for :CO:.

BUFFER — A POINTER to a buffer from which data is to be transferred to the open file.

COUNT — A WORD containing the number of bytes to be transferred from the buffer to the file. The value of COUNT is added to MARKER. If this results in MARKER being greater than length, then LENGTH is set equal to MARKER. The number of bytes actually transferred by WRITE is exactly equal to COUNT. Thus, if the buffer length is less than COUNT, memory locations following the buffer are written to the file.

STATUS — A POINTER to a WORD to which the monitor will return a status code in the vent of a nonfatal ISIS error. The errors and their numbers are listed in Appendix A.

Example

The following PL/M-86 program segments illustrate using the WRITE routine to transfer 128 bytes from a buffer in memory to :CO:.

```
WRITE:
    PROCEDURE (AFTN, BUFFER, COUNT, STATUS) EXTERNAL;
      DECLARE (AFTN, COUNT) WORD;
      DECLARE (BUFFER, STATUS) POINTER;'
    END WRITE;
      .
      .
      .
    DECLARE BUFFER(128) BYE;
    DECLARE STATUS$VALUE WORD;
      .
      .
      .
    CALL WRITE (0, @BUFFER, 128, @STATUS$VALUE);
      .
      .
      .
```

SEEK ROUTINE

A call to the SEEK routine changes or returns the value of the MARKER associated with an open file. The SEEK call can only be used with a file open for updating or reading. The MARKER can be changed in four ways: moved forward, moved backward, moved to a specific location, or moved to the end of the file. A nonfatal ISIS error occurs if there is an attempt to SEEK beyond the end of a file open for reading.

The arguments that must be passed in a call to SEEK are the following:

AFTN — A WORD containing the active file number of a file that is open for reading or updating. The AFTN value was returned by a previous call to the OPEN routine.

MODE — A WORD containing an indicator of the action that is to be performed on MARKER. The next two arguments, BLOCKNO and BYTENO, are directly affected by MODE. The possible values of MODE are as follows:

- A value of 0 specifies that ISIS-II is to return (in BLOCKNO and BYTENO) the current block-number and byte-number values, respectively, that determine the value of MARKER. The current location of MARKER can be computed from these values by the formula

  MARKER = 128*(block-number MODULO 32768)+byte-number.

- A value of 1 specifies that MARKER is to be moved backward by the value

  N = 128*(block-number MODULO 32768)+byte-number.

  If the resulting value of MARKER is negative, MARKER is set to 0 and a nonfatal error occurs.

- A value of 2 specifies that MARKER is to be set to the value

  N = 128*(block-number MODULO 32768)+byte-number.

  If N is zero, MARKER is moved to the beginning of the file. If the new value of MARKER is greater than LENGTH, enough ASCII nulls (00H) are appended to the file that LENGTH equals MARKER.

- A value of 3 specifies that MARKER is to be moved forward by the value

  N = 128*(block-number MODULO 32768)+byte-number.

  If the resulting value of MARKER is greater than LENGTH, enough ASCII nulls (00H) are appended to the file that LENGTH equals MARKER.

- A value of 4 specifies that MARKER be set to LENGTH. This moves MARKER to the end of the file. The arguments BLOCKNO and BYTENO are not used or affected when MODE is 4.

BLOCKNO A POINTER to a WORD containing the number of a block in a the file. A block is equivalent to 128 bytes.

BYTENO A POINTER to a WORD containing the number of a byte within a block in the file. The byte number may be greater than 128.

STATUS A POINTER to a WORD to which the monitor will return a status code in the event of a nonfatal ISIS error. The errors and their numbers are listed in Appendix A.

Example

The following PL/M-86 program segments illustrate using the SEEK routine to obtain the current BLOCKNO and BYTENO components of MARKER, for a file whose AFTN is in FILE$AFTN.

```
SEEK:
    PROCEDURE (AFTN, MODE, BLOCKNO, BYTENO, STATUS) EXTERNAL;
      DECLARE (AFTN, MODE) WORD;
      DECLARE (BLOCKNO, BYTENO, STATUS) POINTER;
    END SEEK;
      .
      .
      .
    DECLARE FILE$AFTN WORD;
    DECLARE BLOCK$NUMBER WORD;
    DECLARE BYTE$NUMBER WORD;
    DECLARE STATUS$VALUE WORD;
      .
      .
      .
    CALL SEEK(FILE$AFTN, 0, @BLOCK$NUMBER, @BYTE$NUMBER,
    @STATUS$VALUE);
      .
      .
      .
```

RESCAN ROUTINE

A call to the RESCAN routine has no effect. It is provided only for compatibility with ISIS.

The arguments that must be passed in a call to RESCAN are the following:

AFTN — A WORD containing the active file number of a file that is open for line-edited input (with echo file AFTN specified).

STATUS — A POINTER to a WORD to which nothing is returned.

CLOSE ROUTINE

A call to the CLOSE routine deletes the tables and buffers that were allocated when the specified file was opened. All files should be closed when I/O processing is complete.

The arguments that must be passed in a call to CLOSE are the following:

AFTN — A WORD containing the active file number of an open file. The AFTN was returned by a previous call to the OPEN routine.

STATUS — A POINTER to a WORD to which the monitor will return a status code in the event of a nonfatal ISIS error. The errors and their numbers are listed in Appendix A.

Example

The following PL/M-86 program segments illustrate using the CLOSE command to close a file whose AFTN is in FILE$AFTN.

```
CLOSE:
    PROCEDURE (AFTN, STATUS) EXTERNAL;
      DECLARE AFTN WORD;
      DECLARE STATUS POINTER;
    END CLOSE;
      .
      .
      .
    DECLARE FILE$AFTN WORD;
    DECLARE STATUS$VALUE WORD;
      .
      .
      .
    CALL CLOSE(FILE$AFTN, @STATUS$VALUE);
      .
      .
      .
```

DELETE ROUTINE

A call to the DELETE routine removes the specified file from its disk. The space occupied by the file is released and becomes available for use by another file.

The arguments that must be passed in a call to DELETE are the following:

FILENAME A POINTER to an ASCII string containing the name of the file to be deleted. The file must not be open. The string can contain leading space characters but no embedded space characters. It must be terminated by a character other than a letter, digit, colon (:), or period (.). A space may be used as a terminating character.

STATUS A POINTER to a WORD to which the monitor will return a status code in the event of a nonfatal ISIS error. The errors and their numbers are listed in Appendix A.

Example

The following PL/M-86 program segments illustrate deleting a file whose name is OLD.FIL. OLD.FIL resides in drive 1.

```
DELETE:
    PROCEDURE (FILENAME, STATUS) EXTERNAL;
      DECLARE(FILENAME, STATUS) POINTER,
    END DELETE;
      .
```

Example (continued)

```
        .
        .
    DECLARE STATUS$VALUE WORD;
        .
        .
        .
    CALL DELETE(@(':F1:OLD.FIL '), @STATUS$VALUE);
```

RENAME ROUTINE

A call to the RENAME routine changes the name of a disk file.

The arguments that must be passed in a call to RENAME are the following:

OLDNAME A POINTER to an ASCII string containing the old file name. The string can contain leading space characters but no embedded space characters. It must be terminated by a character other than a letter, digit, colon (:), or period (.). A space may be used as a terminating character.

NEWNAME A POINTER to an ASCII string containing the new file name. The string can contain leading space characters but no embedded space characters. It must be terminated by a character other than a letter, digit, colon (:), or period (.). A space may be used as a terminating character. The device portion of the name must be the same as that in the old name.

STATUS A POINTER to a WORD to which the monitor will return a status code in the event of a nonfatal ISIS error. The errors and their numbers are listed in Appendix A.

Example

The following PL/M-86 program segments illustrate using the RENAME routine to change the name of a file from BAD.NAM to GOOD.NAM. The file resides in drive 1.

```
RENAME:
    PROCEDURE (OLDNAME, NEWNAME, STATUS) EXTERNAL;
      DECLARE (OLDNAME, NEWNAME, STATUS) POINTER;
    END RENAME;
        .
        .
        .
    DECLARE STATUS$VALUE WORD;
        .
        .
        .
    CALL RENAME(@(':F1:BAD.NAM '), @(':F1:GOOD.NAM '), @STATUS$VALUE);
```

Example

The following PL/M-86 program segments illustrate using the CLOSE command to close a file whose AFTN is in FILE$AFTN.

```
CLOSE:
    PROCEDURE (AFTN, STATUS) EXTERNAL;
      DECLARE AFTN WORD;
      DECLARE STATUS POINTER;
    END CLOSE;
      .
      .
      .
    DECLARE FILE$AFTN WORD;
    DECLARE STATUS$VALUE WORD;
      .
      .
      .
    CALL CLOSE(FILE$AFTN, @STATUS$VALUE);
      .
      .
      .
```

DELETE ROUTINE

A call to the DELETE routine removes the specified file from its disk. The space occupied by the file is released and becomes available for use by another file.

The arguments that must be passed in a call to DELETE are the following:

FILENAME A POINTER to an ASCII string containing the name of the file to be deleted. The file must not be open. The string can contain leading space characters but no embedded space characters. It must be terminated by a character other than a letter, digit, colon (:), or period (.). A space may be used as a terminating character.

STATUS A POINTER to a WORD to which the monitor will return a status code in the event of a nonfatal ISIS error. The errors and their numbers are listed in Appendix A.

Example

The following PL/M-86 program segments illustrate deleting a file whose name is OLD.FIL. OLD.FIL resides in drive 1.

```
DELETE:
    PROCEDURE (FILENAME, STATUS) EXTERNAL;
      DECLARE(FILENAME, STATUS) POINTER,
    END DELETE;
      .
```

Example (continued)

```
        .
        .
    DECLARE STATUS$VALUE WORD;
        .
        .
        .
    CALL DELETE(@(':F1:OLD.FIL '), @STATUS$VALUE);
```

RENAME ROUTINE

A call to the RENAME routine changes the name of a disk file.

The arguments that must be passed in a call to RENAME are the following:

OLDNAME A POINTER to an ASCII string containing the old file name. The string can contain leading space characters but no embedded space characters. It must be terminated by a character other than a letter, digit, colon (:), or period (.). A space may be used as a terminating character.

NEWNAME A POINTER to an ASCII string containing the new file name. The string can contain leading space characters but no embedded space characters. It must be terminated by a character other than a letter, digit, colon (:), or period (.). A space may be used as a terminating character. The device portion of the name must be the same as that in the old name.

STATUS A POINTER to a WORD to which the monitor will return a status code in the event of a nonfatal ISIS error. The errors and their numbers are listed in Appendix A.

Example

The following PL/M-86 program segments illustrate using the RENAME routine to change the name of a file from BAD.NAM to GOOD.NAM. The file resides in drive 1.

```
RENAME:
    PROCEDURE (OLDNAME, NEWNAME, STATUS) EXTERNAL;
      DECLARE (OLDNAME, NEWNAME, STATUS) POINTER;
    END RENAME;
        .
        .
        .
    DECLARE STATUS$VALUE WORD;
        .
        .
        .
    CALL RENAME(@(':F1:BAD.NAM '), @(':F1:GOOD.NAM '), @STATUS$VALUE);
```

ATTRIB ROUTINE

A call to the ATTRIB routine changes an ISIS-II attribute of a disk file.

The arguments that must be passed in a call to ATTRIB are the following:

FILENAME — A POINTER to a string containing the name of the file whose attribute is to be changed. The string can contain leading space characters but no embedded space characters. It must be terminated by a character other than a letter, digit, colon (:), or period (.). A space may be used as a terminating character.

ATTRIBUTE — A WORD indicating which attribute is to be changed. Possible values are:

- 0 - invisible attribute
- 1 - system attribute
- 2 - write protect attribute
- 3 - format attribute

ONOFF — A WORD indicating whether the attribute is to be set (turned on) or reset (turned off). A value of 1 specifies that the attribute be set, and a value of 0 indicates that it be reset.

STATUS — A POINTER to a WORD to which the monitor will return a status code in the event of a nonfatal ISIS error. The errors and their numbers are listed in Appendix A.

Example

The following PL/M-86 program segments illustrate using the ATTRIB routine to reset the write protect attribute of a file named SCRTCH.PAD. The file resides in drive 1.

```
ATTRIB:
    PROCEDURE (FILENAME, ATTRIBUTE, ONOFF, STATUS) EXTERNAL;
      DECLARE (FILENAME, STATUS) POINTER;
      DECLARE (ATTRIBUTE, ONOFF) WORD;
    END ATTRIB;
      .
      .
      .
    DECLARE STATUS$VALUE WORD;
      .
      .
      .
    CALL ATTRIB(@(':F1:SCRTCH.PAD '), 2, 0, @STATUS$VALUE);
```

LOAD ROUTINE

A call to the LOAD routine loads a located object file from a disk in the Intellec system into iAPX 86, 88 memory. The file must have been created by LOC86. After the file is loaded, control is passed either to the loaded program or to the calling program, as specified in the call.

The arguments that must be passed in a call to LOAD are the following:

FILENAME A POINTER to an ASCII string containing the name of the file to be loaded. The string can contain leading space characters but no embedded space characters. It must be terminated by a character other than a letter, digit, colon (:), or period (.). A space may be used as a terminating character.

BIAS A WORD containing zero. This argument is included for compatibility with ISIS-II. The zero value assures that the argument has no effect.

SWITCH A WORD indicating where control is to be transferred after the load operation. The possible values of SWITCH are the following:

- A value of 0 indicates that control is to be returned to the calling program. If a starting address is available in the loaded file, it is returned to the location pointed to by the ENTRY argument.
- A value of 1 indicates that control is to be passed to the loaded program. This is the load and go option.

ENTRY A POINTER to a POINTER in which the starting address of the loaded program is placed.

STATUS A POINTER to a WORD to which the monitor will return a status code in the event of a nonfatal ISIS error. The errors and their numbers are listed in Appendix A.

Example

The following PL/M-86 program segments illustrate using the LOAD routine to load and start execution of a file named LOADED.FIL. The file resides in drive 1.

```
LOAD:
    PROCEDURE (FILENAME, BIAS, SWITCH, ENTRY, STATUS) EXTERNAL;
      DECLARE (FILENAME, ENTRY, STATUS) POINTER;
      DECLARE (BIAS, SWITCH) WORD;
    END LOAD;
      .
      .
      .
    DECLARE START$ADDRESS POINTER:
```

Example (continued)

```
DECLARE STATUS$VALUE WORD;
  .
  .
  .
CALL LOAD (@(':F1:LOADED.FIL '), 0, 1, @START$ADDRESS,
@STATUS$VALUE);
  .
  .
  .
```

ERROR ROUTINE

A call to the ERROR routine sends an error message to the Intellec system console.

The argument that must be passed in a call to ERROR is the following:

ERRNUM A WORD containing the number of the error that is to be output to the console. Only the values 101 through 199, inclusive, should be employed for user programs; the other error numbers (0-100 and 200-255) are reserved for system programs. The system displays the error in the following format:

ERROR nnn, USER PC mmmm

where nnn is ERRNUM and mmmm is the return address in the calling program. Note that, because the return address given is an 8080 address, it is of little use to an 8086 programmer.

Example

The following program segments illustrate sending error number 147 to the console.

```
ERROR:
    PROCEDURE (ERRNUM) EXTERNAL;
      DECLARE ERRNUM WORD;
    END ERROR;
      .
      .
      .
    CALL ERROR (147)
      .
      .
      .
```

CI ROUTINE

A call to the CI routine returns to the AL register an ASCII character received from the console device. The AX, CX, and DX registers and the CPU condition codes are affected by this operation.

If a Control-S (13H) or Control-Q (11H) character is read, the character is "thrown away" and another is read.

Example

The following PL/M-86 program segments illustrate using the CI routine to obtain a character from the console.

```
CI:
    PROCEDURE BYTE EXTERNAL;
    END CI;
      .
      .
      .
    DECLARE CHAR BYTE;
      .
      .
      .
    CHAR = CI AND 7FH; /* INPUT CHARACTER AND STRIP PARITY BIT */
      .
      .
      .
```

CO ROUTINE

A call to the CO routine transfers a character from the low-order byte of the word on the top of the stack to the console device. The AX, CX, and DX registers and the CPU condition codes are affected by this operation.

Before the character is transmitted, the routine checks to see if a Control-S has been received. If so, it waits until a Control-Q is received before transmitting the character. Therefore, if a Control-S is entered at the console when output is pending. The output is temporarily stopped. Entering a Control-Q resumes output.

The argument that must be passed in a call to CO is the following:

CHARACTER A BYTE containing an ASCII character.

Example

The following PL/M-86 program segments illustrate using the CO routine to transmit a character to the console.

```
CO:
    PROCEDURE (CHARACTER) EXTERNAL;
      DECLARE CHARACTER BYTE;
    END CO;
      .
      .
      .
    DECLARE CHAR BYTE;
      .
      .
      .
    CALL CO(CHAR);
      .
      .
      .
```

ISIS ROUTINE

In the APXIOS.LIB library (for the "small" control), a PUBLIC called ISIS is provided as an alternate means of utilizing most of the routines described earlier in this chapter. For most users, the routines already described are easier to use. However, for certain users with PL/M-80 programs that are being recompiled under PL/M-86 with the "small" control, the ISIS routine allows for compatibility of those programs with PL/M-86.

The ISIS routine is not included in either the APXIOL.LIB or the APXIOC.LIB libraries.

The routines supported by the ISIS routine have already been described. To call these routines using ISIS, the following arguments must be passed:

FUNCTION — A WORD containing a code for the specific routine that is to be called. The routines supported, with their associated numerical values, are the following:

0 - OPEN
1 - CLOSE
2 - DELETE
3 - READ
4 - WRITE
5 - SEEK
6 - LOAD
7 - RENAME
9 - EXIT
10 - ATTRIB
11 - RESCAN
12 - ERROR

PARAM$BLOCK — A WORD containing the address of a structure. The structure contains the arguments required by the particular routine being called. The arguments must be placed in the structure in the order in which the arguments are listed earlier in this chapter.

Example

The following PL/M-86 program segments illustrate using the ISIS command to make a call to the WRITE command. The result of the call is that the word "HELLO" is sent to the console, followed by a carriage return and a line feed.

```
ISIS:
    PROCEDURE (FUNCTION, PARAM$BLOCK) EXTERNAL;
      DECLARE (FUNCTION, PARAM$BLOCK) WORD;
    END ISIS;
      .
      .
      .
    DECLARE STAT WORD;
    DECLARE LBUFF(100) BYTE INITIAL ('HELLO', 0DH, 0AH);
    DECLARE PBLOCK STRUCTURE
                  (AFTN   WORD,
                   BUFFER WORD,
                   COUNT  WORD,
                   STATUS WORD)
            INITIAL(0, @LBUFF, 7, @STAT);
      .
      .
      .
    CALL ISIS(4, @PBLOCK);
      .
      .
      .
```

Note that, in this example, WORD can be replaced by either ADDRESS or POINTER, and the "at" sign (@) can be replaced by a period (.).

EXIT ROUTINE

A call to the EXIT routine causes a jump to the iAPX 86, 88 monitor command level. EXIT does not close any files.

Example

```
EXIT:
    PROCEDURE EXTERNAL;
    END EXIT;
      .
      .
      .
    CALL EXIT;
      .
      .
      .
```

CHAPTER 5. PROGRAMMING INFORMATION

A variety of information, concerning memory organization, single-stepping, initialized contents of iAPX 86, 88 CPU registers, USART and interrupt controller initialization, interrupt servicing, instruction breakpoints and NPX initialization, is discussed in this chapter.

MEMORY ORGANIZATION

The RAM memory space the monitor uses varies depending on the configuration selected. If NPX support is configured into the monitor, RAM locations 0 to 07FFH are used for monitor operations. However, if NPX support is not configured into the monitor, only RAM locations 0-06FFH are used for monitor operations.

The EPROM memory space the monitor uses also varies depending on the configuration used. All default configurations use EPROM locations 0FC000H to 0FFFFFH, except the default configuration for the iSBC 88/40 board. The default configuration for the iSBC 88/40 board uses EPROM locations 0FD000H to 0FFFFFH, which allows the use of an EEPROM at 0FC000H.

NOTE

Programs cannot be located with the default starting address of 200H. You must use either the following LOC86 or equivalent controls when locating programs:

RESERVE (0H to 7FFH) or
ADDRESSES (SEGMENTS(CODE(800H)))

Memory locations 0H to 3FFH are reserved for 256 four-byte interrupt vectors, numbered 0 to 255. Vectors 0 through 4 have dedicated hardware functions. In particular, vectors 1, 2, and 3 are used by the monitor for single-stepping, non-maskable interrupts, and execution breakpoints, respectively. Vectors 5 through 31 are reserved for future use by Intel. Vectors 32 through 39 are dedicated to interrupt levels 0 through 7, respectively, on the 8259A Programmable Interrupt Controller. The interrupts dedicated to the 8259A PIC and interrupt vectors 40-255 are available to your programs.

vectors 1 — single step.
2 - nmi
3 - execution breakpoints

During start-up, upon reset, and when a main module is loaded by the L or R commands, the stack pointer is set to 428H. The 40 bytes devoted to the user stack are at locations 400H through 427H.

400H – 427H user stack

SPECIAL CONSIDERATIONS WHEN STEPPING

When using hardware single stepping, you should be aware of several special properties of stepping. Hardware single stepping occurs when using the single step (N) command. It also occurs when using the go (G) command, either with memory breakpoints or with an execution breakpoint set at the program's starting address. The following should be considered by a user who employs hardware single stepping:

- There must be 12 bytes of stack available for the exclusive use of the monitor. This need can be satisfied in either of two ways, depending on whether the program calls any of the library routines described later in this chapter. If any such routines are used, 12 extra bytes are automatically added to the stack during linking. If the program does not call any library routines, the user must arrange for the inclusion of the extra bytes by means of the appropriate LOC86 command.

- Because the monitor makes extensive use of the stack when single stepping, problems can occur when stepping through stack initialization code. If the stack is initialized in either of two special ways, the monitor treats the initializing code sequences as a single instruction, thereby avoiding trouble. The two code sequences that the monitor treats in this manner are

```
MOV SS,CS:WORD PTR STACKSEG
MOV SP,OFFSET STACKSIZE
```

  and

```
MOV SS,ES:WORD PTR STACKSEG
MOV SP,ES:WORD PTR STACKPTR
```

  The former method is used by the PL/M-86 compiler. When programming in assembly language, one of these methods should be used to initialize the stack.

- The monitor assumes that only one-byte instructions will be REPeated. An attempt to REP longer instructions will fail. REPed instructions, when stepped, are printed only once, even if they are executed multiple times.

- Single stepping does not automatically disable interrupts, so if your system employs hardware interrupts you might find yourself stepping through interrupt servicing routines. See the "Q" option in chapter 3 under the description of the "N" command.

iAPX 86, 88 CPU REGISTER INITIALIZATION

When an iAPX 86, 88-based board is powered-on or reset, the monitor initializes the iAPX 86, 88 registers with the following values:

CS = 0000
SS = 0000
DS = 0000
IP = 0000
FL = 0000
SP = 0428H

USART INITIALIZATION

When the processor board is powered-on or reset, the monitor automatically programs the 8251A USART for interfacing with the console device. If there is a parallel interface between the Intellec system and the processor board, the USART is not used, although it is still programmed. The following indicate the state of the USART after programming:

a. Mode:
   1 stop bit at 150-9600 baud
   Parity disabled
   8 bit character length
   Baud rate factor of 16X

b. Command:
   Request-To-Send
   Error reset
   Received enabled
   Data-Terminal-Ready at 150-9600 baud
   Transmitter enabled
   Non-hunt mode

NOTE

Counter 2 of the 8253 Programmable Interval Timer is used to establish the baud rate for serial communication. Care must be used when modifying the USART mode and command, because the monitor's device drivers depend on the conditions just listed for proper operation. The mode of Counter 2 should not be modified.

INTERRUPT SERVICING

When the processor board is powered on or is reset the monitor sets the following:

1) the divide error interrupt vector
2) the single step interrupt vector
3) the non-maskable interrupt vector
4) the one-byte trap instruction interrupt vector
5) the overflow interrupt vector

The monitor also sets interrupt vectors 32 through 39.

The monitor initializes the 8259A Programmable Interrupt Controller as follows:

8086 edge mode
4 bytes per interrupt level
interrupt vector for 8259A begins at 800H
Fully nested
Edge-triggered
Buffered or non-buffered(configuration option)
Normal end-of-interrupt
No slaves
Level 0 has highest priority
80130 - no ICW5 or ICW6 needed
All 8259A interrupt levels are masked.

Interrupts are disabled during user/monitor command interaction so that pending interrupts will not interfere with program examination. Your program's interrupt state is restored on exiting the monitor by way of the Go (G) or Single-step (N) command.

If you want to service interrupts through the 8259A controller, you must modify the appropriate interrupt vectors. To prepare for interrupts coming in on 8259A level n (where n is between 0 and 7, inclusive), the address of the interrupt servicing routine is placed in vector 32 + n. This is accomplished by placing the IP value at

80H + 4*n

and the CS value at

82H + 4*n.

If an interrupt arrives and the corresponding vector has not been modified, the message

Break at hex-value:hex-value

is sent to the console. The hex values are the values of CS and IP at the time of the interrupt.

INSTRUCTION BREAKPOINTS

If an INT 3 instruction is placed in a program, the monitor is reentered when that instruction is executed. The monitor saves all registers and then prints the message

Break at hex-value:hex-value

where the hex values indicate the location of the INT3 instruction. The monitor then issues a prompt. Program execution may be resumed with the Go (G) or Single Step (N) command.

Causing the Non-maskable pin on the 8086 or the 8088 to go active will generate a type 2 (non-maskable) interrupt. Generation of this interrupt causes the processor's execution to be halted at the current CS:IP and then causes the monitor to be reentered. The monitor saves all registers and then prints the message:

BREAK at hex-value:hex-value

where the hex values indicate the values of the CS and IP registers, respectively, where execution was halted.

Following this display, you may issue any desired monitor commands. Program execution may be resumed with the Go (G) or Single Step (N) command.

NPX INITIALIZATION

The monitor uses an initialization sequence for the NPX that requires no jumper modification of the iAPX 86, 88-based board whether the NPX is installed on it or not. NPX users should refer to the Application Note, Getting Started With the Numeric Data Processor, for an example of this initialization sequence and other NPX details.

EEPROM TIMER INITIALIZATION

When the monitor initializes counter 2 of an on-board PIT to support writing to EEPROM memory, it programs the counter in mode 1 with a count sufficient to generate a 14 mSec. programming pulse. See the iSBC 88/40 HARDWARE REFERENCE MANUAL for hardware details.

CHAPTER 6. CONFIGURING THE iAPX 86, 88 MONITOR

The iSBC 957B package provides the capability for you to configure the iAPX 86, 88 monitor to run on any iAPX 86, 88-based board. Monitor configuration involves selecting characteristics and options you need in the iAPX 86, 88 monitor and specifying information about the iAPX 86, 88-based board the monitor is to run on. You perform these operations by making modifications to an Intel-supplied iAPX 86, 88 monitor configuration source file or by creating a configuration source file that describes your iAPX 86, 88-based board. The monitor configuration source files are assembly language source modules, which consist of calls to the monitor configuration macros. The iSBC 957B package provides four monitor configuration source files, one for each of the supported Intel single board computers, listed in Table 6-1.

Table 6-1. iAPX 86, 88 Monitor Configuration Source Files

SINGLE BOARD COMPUTER	CONFIGURATION FILE
iSBC 86/12A	CF8612.A86
iSBC 86/05	CF8605.A86
iSBC 88/40	CF8840.A86
iSBC 88/25	CF8825.A86

The iSBC 957B package includes a set of four 2732 EPROMs which support the iSBC 86/12A and iSBC 86/05 board and a set of three 2732 EPROMs which support the iSBC 88/40 board. The iSBC 957B package does not include a default monitor, burned into EPROM, which supports the iSBC 88/25 board. Instead, there is a submit file, CF8825.CSD, and a configuration source file, CF8825.A86, to generate such a monitor configuration.

If you want the monitor to run on a different hardware configuration, or if you want to change some of the characteristics of the monitor, you must modify the configuration source file for the particular iAPX 86, 88-based board you are using, assemble it, link it with the rest of the monitor, locate it and burn the located code into EPROM.

The file, CF957B.MAC, which is contained on the diskettes provided in the iSBC 957B package, contains definitions of all of the monitor configuration macros. A configuration source file must contain an $INCLUDE statement for CF957B.MAC.

CONFIGURATION SOURCE FILE CONTENTS

A configuration source file contains macro calls which identify the charateristics of the monitor, the Intellec communication link, the iAPX 86, 88-based board, the 8087 Numeric Processor Extension (NPX) and the Bootstrap Loader option. When configuring a monitor with no Bootstrap Loader option selected, a configuration source file consists of a set of nine macro calls. When configuring a monitor with the Bootstrap Loader option selected, a configuration source file consists of twelve or more macro calls depending on the number of different devices you wish to bootstrap from.

A configuration source file always contains calls to the following nine macros in the following order:

```
CPU
MAX_BAUD_RATE_COUNT
BAUD_RATE
BAUD_RATE_TIMER
EXTRA_TIMER
SERIAL_PORT
PARALLEL_PORT
INTERRUPT_CONTROLLER
NPX
```

If the Bootstrap Loader option is to be selected, the configuration source file contains calls to the nine macros listed above and then contains calls to the following three additional macros:

```
BOOTSTRAP
DEVICE
END_BOOTSTRAP
```

The configuration macros perform three distinct types of functions.

1. The CPU, MAX_BAUD_RATE_COUNT, BAUD_RATE, BAUD_RATE_TIMER, EXTRA TIMER, SERIAL_PORT, PARALLEL_PORT and INTERRUPT_CONTROLLER macros define hardware and software attributes of the iAPX 86, 88-based board used.

2. The NPX macro configures support for the NPX into or out of the monitor. NPX support includes the ability to examine and/or modify the NPX state, and the ability to examine and/or modify NPX (real, integer or BCD) data types.

3. The BOOTSTRAP, DEVICE and END_BOOTSTRAP macros configure the iRMX 86 or 88 Bootstrap Loader into the monitor. If these macros are not called, the Bootstrap Loader is not configured into the monitor.

The MAX_BAUD_RATE_COUNT, BAUD_RATE, BAUD_RATE_TIMER and SERIAL_PORT macros all provide information to define the serial communication link. Their parameters interact with each other as defined in the sections below describing each macro.

Invalid parameters will generate a syntax error when assembling the configuration source file. The statement causing the syntax error and an error message defining the invalid parameter are output, as in the following example:

```
                                31 +3  ERROR -- 8087 is an invalid CPU type
***  _____________________________________^
*** ERROR #1, LINE #32, SYNTAX ERROR
```

The BAUD_RATE_TIMER, EXTRA_TIMER, SERIAL_PORT, PARALLEL_PORT and INTERRUPT_CONTROLLER macros may have a type specification of "NONE" if there is none of those devices on the iAPX 86, 88-based board. If the type specification is "NONE", then no other parameters need to be supplied in the macro call. An example of this type of macro call is:

```
%EXTRA_TIMER(NONE)
```

THE MONITOR CONFIGURATION MACROS

The following sections describe the monitor configuration macros in detail.

THE CPU MACRO

This macro allows you to specify the type of CPU your iAPX 86, 88-based board uses and whether or not the board supports bus vectored (cascaded) interrupts, that is, whether the 8259A should be programmed in buffered mode.

```
%CPU (cpu_type, bus_vectored_interrupts)
```

where

cpu_type	A WORD value that indicates the type of CPU your iAPX 86, 88-based board uses. The two valid types are 8086 and 8088.
bus_vectored_interrupts	A character that indicates whether or not your processor board is capable of bus vectoring interrupts (cascading interrupts). Y indicates your processor board can bus vector interrupts, N indicates it cannot.

Error Messages

ERROR - <type> is an invalid CPU type

THE MAX_BAUD_RATE_COUNT MACRO

This macro provides the monitor with a value that can be used to program the programmable interval timer (PIT) used for baud rate generation to generate a baud rate of 9600 baud. The input clock frequency of the baud rate timer is used to derive this value. The iSBC 86/12A, iSBC 86/05 and iSBC 88/25 single board computers and the iSBC 351 serial multimodule all come default configured to provide an input clock frequency of 1.2288 megahertz to counters 0 and 2. If you use counters 0 or 2 of the above mentioned boards, a value of 8 is required as input to this macro.

If your iAPX 86, 88-based board's PIT has a input clock frequency other than 1.2288 megahertz, you must compute a new value and then call this macro with the new value to set the limits on the baud rate attributes of the Serial Port. The BAUD_RATE macro will use the value provided here to derive a count to generate the baud rate specified in the BAUD_RATE macro. The format of the call to this macro is as follows:

```
%MAX_BAUD_RATE_COUNT (count)
```

where

count	A WORD value that, if loaded into the timer register, generates the maximum supported baud rate of 9600. The value that you enter for this parameter depends on the input clock frequency to your system's PIT (refer to the following paragraphs for computing this value).

Error Messages

```
ERROR - Max Baud Rate Count must be greater than 1
```

To derive the value to use for the count parameter, you must first determine the clock input frequency to the PIT (in hertz). Then substitute this frequency into the following equation:

(1) result = (clock frequency in hertz) / (9600 X 16)

Then substitute "result" from equation 1 into the following equation:

(2) fraction = result - INT(result)

where INT(result) is an integer obtained by truncating the fractional portion of "result". If "fraction" from equation 2 is greater than or equal to 0.5, then:

(3) count = INT(result) + 1

error_fraction = 1.0 - fraction

If "fraction" is less than 0.5, then:

(4) count = INT(result)

error_fraction = fraction

Before placing "count" from equation 3 or 4 into the %MAX_BAUD_RATE_COUNT call, you should first determine the percentage of error in this value.

You do this by evaluating the following expression:

(5) % error = (error_fraction / count) X 100

If the percentage of error is less than 3%, you can use any monitor-supported baud rate (which you later specify in the macro, BAUD RATE, described later in this chapter). However, if the percentage of error is 3% or greater, you will have to perform the following additional computations.

First, determine the desired baud rate of the terminal. Substitute this value for the 9600 in equation 1 and recompute the value of "count" (equations 1 through 4). Again determine the percentage of error (equation 5). If the error is less than 3% with the new baud rate, you can use the monitor with that new baud rate (and optionally specify it in the BAUD_RATE macro). However, if the percentage of error is still 3% or greater, the combination of desired baud rate and clock frequency is unacceptable to the monitor. You will have to change one or the other. After doing this, recompute the percentage of error to verify that it falls below the 3% level.

Regardless of the baud rate you eventually choose, use the "count" value as originally computed (with the 9600 value) as input to the MAX_BAUD RATE_COUNT macro.

THE BAUD_RATE MACRO

This macro allows you to specify the baud rate of the terminal used in a serial communication link. The format of this call is as follows:

```
%BAUD_RATE (rate)
```

where

rate	A WORD value indicating the baud rate of the terminal connected to the serial port of the processor board. Specify one of the following rates:

9600
4800
2400
1200
600
300
150
0

If rate is 0 you can use either the parallel or the serial Intellec interface or a stand alone terminal serial interface, depending on the hardware configuration. If rate is 0 and the interface is serial, the monitor will perform a baud rate scan to determine the correct baud rate for the terminal connected. However, if rate is non-zero, the monitor assumes that only a stand alone serial terminal interface exists. The monitor will initialize the hardware for the specified baud rate and will sign-on when power-on occurs or when the system is reset. To use a different baud rate, once it is set in this manner, requires changing the parameter "rate" in this macro call, reconfiguring the monitor, and burning it into another set of EPROMs.

This macro will compute a count, based on the value supplied in the MAX BAUD_RATE_COUNT macro and the baud rate specified here, that will be used to program the baud rate counter.

Error Messages

ERROR - invalid Baud Rate

THE BAUD_RATE_TIMER MACRO

This macro allows you to specify attributes of an optional Programmable Interval Timer (PIT) on the processor board. If the type specified is NONE, the monitor assumes that a serial communication link will not exist; only a parallel link may be used. In addition, if the type is NONE, the type parameter in the SERIAL_PORT macro must also be NONE. This is because the PIT described in this macro and the USART described by the SERIAL_PORT macro function together as a unit; the existence of one requires the existence of the other. The format of this call is as follows:

```
%BAUD_RATE_TIMER  (type, base_port, port_delta, baud_counter)
```

where

type	A value that describes the device used for baud rate generation. The valid types are 8253, 8254, 80130 and NONE.
base_port	A WORD value that specifies the numerically lowest valued port on the baud rate timer.

port_delta	A BYTE value that specifies the number of bytes between the ports in the Timer. A port_delta of 1 yields port addresses of base_port + 0, base_port + 1, base_port + 2 and so on. A port_delta of 2 yields port addresses of base_port + 0, base_port + 2, base_port + 4 and so on. 0 is not a valid value for port_delta.
baud_counter	The BYTE value of the timer on the PIT connected to the USART used for baud rate generation. Valid values are 0, 1 and 2.

Error Messages

```
ERROR - Invalid port delta for the Baud Rate Timer
ERROR - Baud Rate Counter is not 0, 1 or 2
ERROR - 2 is the only valid 80130 Baud Rate Timer
ERROR - <type> is an invalid Baud Rate Timer type
```

THE EXTRA_TIMER MACRO

This macro allows you to specify information about an optional (extra) Programmable Interval Timer (PIT) which is not used for baud rate generation. An example of this situation would be when the iSBC 88/40 board has an iSBX 351 serial multimodule installed on it for serial I/O leaving the iSBC 88/40 on-board PIT unused. To prevent the counters on the on-board PIT from inadvertantly counting, the PIT counters are programmed into a non-active state by the monitor.

If the on-board PIT is unused and the processor board it is on has the capability of writing to EEPROMs (as is the case with the iSBC 88/40 board), counter 2 of the on-board PIT may be used for EEPROM programming pulse generation. This is accomplished by following the hardware configuration instructions described in Chapter 2 and by specifying the appropriate value in the "eeprom" parameter described below.

The format of the call is as follows:

```
%EXTRA_TIMER (type, base_port, port_delta, eeprom)
```

where

type	A value that describes the device that exists as an extra timer. The valid types are 8253, 8254, 80130 and NONE.
base_port	A WORD value that specifies the numerically lowest valued port on the extra timer.

port_delta	A BYTE value that specifies the number of bytes between the ports in the timer. A port_delta of 1 yields port addresses of base_port +0, base_port + 1, base_port + 2 and so on. A port_delta of 2 yields port addresses of base_port + 0, base_port + 2, base_port + 4 and so on. 0 is not a valid value for port_delta.
eeprom	A character that indicates whether Counter 2 of the extra timer will be used for generating EEPROM programming pulses. Y indicates Counter 2 will be used for this purpose, N indicates it will not.

Error Messages

ERROR - invalid port delta for the Extra Timer
ERROR - <type> is an invalid Extra Timer type

THE SERIAL_PORT MACRO

This macro allows you to specify information about the optional device (USART) used for serial communication. If the type specified is "NONE", the type specified in the BAUD_RATE_TIMER macro must also be "NONE". This is because the USART described in this macro and the PIT described by the BAUD_RATE_TIMER macro function together as a unit; the existence of one requires the existence of the other. The format of the call is as follows:

```
%SERIAL_PORT (type, base_port, port_delta)
```

where

type	A value that describes the USART used as a serial port. The valid types are 8251A or NONE.
base_port	A WORD value that specifies the numerically lowest valued port on the USART.
port_delta	A BYTE value that specifies the number of bytes between the ports in the USART. A port_delta of 1 yields port addresses of base_port + 0, base_port + 2 and so on. A port_delta of 2 yields port addresses of base_port + 0, base_port + 2, base port + 4 and so on. 0 is not a valid value for port_delta.

Error Messages

ERROR - invalid port delta for the Serial Port
ERROR - <type> is an invalid Serial Port type

THE PARALLEL_PORT MACRO

This macro allows you to specify information about the optional device used for parallel communication. If the type specified is "NONE", the monitor assumes that no parallel interface exists; that only a serial interface will be used. The format is as follows:

```
%PARALLEL_PORT (type, base_port, port_delta)
```

where

type	A value that describes the device (Programmable Parallel Interface - PPI) used as the parallel port for a parallel interface. The valid types are 8255A or NONE.
base_port	A WORD value that specifies the numerically lowest valued port on the PPI.
port_delta	A BYTE value that specifies the number of bytes between the ports in the PPI. A port_delta of 1 yields port addresses of base_port + 0, base_port + 2 and so on. A port_delta of 2 yields port addresses of base_port + 0, base_port + 2, base port + 4 and so on. 0 is not a valid value for port_delta.

Error Messages

```
ERROR - invalid port delta for the Parallel Port
ERROR - <type> is an invalid Parallel Port type
```

THE INTERRUPT_CONTROLLER MACRO

This macro allows you to specify information about the optional device used to handle external interrupts on the iAPX 86, 88-based board. If the type is "NONE", the monitor assumes that no Interrupt Controller exists on your iAPX 86, 88-based board and does not initialize interrupt vectors 32 - 39. The format for the call is as follows:

```
%INTERRUPT_CONTROLLER (type, base_port, port_delta)
```

where

type	A value that describes the device used to handle external interrupts. The valid types are 8259A, 80130 and NONE.
base_port	A WORD value that specifies the numerically lowest valued port on the Interrupt Controller.

port_delta — A BYTE value that specifies the number of bytes between the ports in the Interrupt Controller. A port_delta of 1 yields port addresses of base_port + 0, base_port + 2 and so on. A port_delta of 2 yields port addresses of base_port + 0, base_port + 2, base_port + 4 and so on. 0 is not a valid value for port_delta.

Error Messages

ERROR - invalid port delta for the Interrupt Controller
ERROR - <type> is an invalid Interrupt Controller type

THE NPX MACRO

This macro allows you to specify whether the monitor will contain NPX support. NPX support includes the capability to display and/or modify the NPX state, NPX stack registers, status, control or tag words and NPX data types. The format for the call is as follows:

```
%NPX (support)
```

where

support — A character which, if Y, indicates that the monitor should include modules to provide NPX support. Approximately 3500 bytes of code and 256 bytes of data are added to the monitor when this option is selected. If the character is N, no NPX support will be provided.

A monitor configured with NPX support in it will function regardless of whether or not an NPX is actually on the processor board. In the absence of an NPX, any attempt to issue NPX support commands to a monitor with NPX support configured in will result in the command being aborted and the error message "NPX Unavailable" be printed. Any attempt to use NPX support commands in a monitor configured without NPX support will always generate the "NPX Unavailable" error.

Error Messages

NONE

BOOTSTRAP LOADER OPTION

The BOOTSTRAP, DEVICE and END_BOOTSTRAP macros form the Bootstrap Loader option and will be discussed first as a unit and then will be described individually. If you are also a user of the iRMX 86 Operating System or the iRMX 88 Real-Time Multitasking Executive, you may configure the Bootstrap Loader first stage into the iAPX 86, 88 monitor. When the Bootstrap Loader option is selected during monitor configuration, the "B" command becomes a valid command, capable of bootstrapping either the iRMX 86 or iRMX 88 system or, alternatively, an application into iAPX 86, 88 memory. After loading the selected code, control passes from the monitor to the code that was loaded (bootstrapped) and execution begins. If the Bootstrap Loader option was not selected, the "B" command is invalid and entering a "B" will cause the error message "Bad Command" to be printed. If the BOOTSTRAP macro is invoked, one or more DEVICE macros and one END BOOTSTRAP macro must be invoked. Selecting this option will cause the Bootstrap Loader to be configured in the "MANUAL" mode. This is the only mode available to iSBC 957B package users. If you are an iRMX 86 user, refer to the section on configuring the Bootstrap Loader in the iRMX 86 CONFIGURATION GUIDE for details. If you are an iRMX 88 user, refer to the iRMX 86 INTERACTIVE CONFIGURATION UTILITIES manual for details.

THE BOOTSTRAP MACRO

This macro allows you to specify whether you want to select the Bootstrap Loader option. To select the Bootstrap Loader option, invoke the BOOTSTRAP macro. The format is as follows:

```
%BOOTSTRAP
```

To indicate you do not wish the Bootstrap Loader option, do not invoke the BOOTSTRAP macro. The format for this is as follows:

```
;BOOTSTRAP
```

Error Messages

NONE

THE DEVICE MACRO

This macro allows you to specify the device(s) and unit(s) that are to be included in the Bootstrap Loader device table. These are the units from which Bootstrap Loader can read. At least one invocation of this macro must occur if the BOOTSTRAP and END-BOOTSTRAP macros are invoked. The format for the call is as follows:

```
%DEVICE (name, unit, device_init, device_read)
```

where

name	The name of the device
unit	The unit number on the device
device_init	The address of the device initialization procedure of the device driver.
device_read	The address of the device read procedure of the device driver.

For more details on the meaning of these parameters, iRMX 86 users should refer to the iRMX 86 CONFIGURATION GUIDE.

Error Messages

ERROR - the BOOTSTRAP macro must be invoked before the DEVICE macros

THE END_BOOTSTRAP MACRO

This macro signals the end of the Bootstrap Loader option. It must be invoked if the BOOTSTRAP macro was invoked. The format for the call is:

```
%END_BOOTSTRAP
```

If the Bootstrap Loader option was not selected, the format for this macro is

```
;END_BOOTSTRAP
```

Error Messages

NONE

SAMPLE MONITOR CONFIGURATIONS

The following is an example of the macro calls used to describe an iAPX 86, 88-based board. The Intel-supplied version of CF8840.A86, describing the iSBC 88/40 board, is used for this purpose.

```
$INCLUDE(:F1:CF957B.MAC)

%CPU(8088, N)
%MAX_BAUD_RATE_COUNT(8)
%BAUD_RATE(0)
%BAUD_RATE_TIMER(8253, 090H, 2, 2)
%EXTRA_TIMER(8253, 0D0H, 2, Y)
```

```
%SERIAL_PORT(8251A, 080H, 2)
%PARALLEL_PORT(8255A, 0C8H, 2)
%INTERRUPT_CONTROLLER(8259A, 0C0H, 2)
%NPX(N)
```

Below is an example of the macro calls used to select the Bootstrap Loader option. This example selects the two devices expected by all default Bootstrap Loader configuration submit files. The example could be used to configure a monitor, with the Bootstrap Loader option selected, for any iAPX 86, 88-based board.

```
%BOOTSTRAP
    %DEVICE(f0, 0, deviceinit204, deviceread204)
    %DEVICE(f1, 1, deviceinit204, deviceread204)
    %DEVICE(f2, 2, deviceinit204, deviceread204)
    %DEVICE(f3, 3, deviceinit204, deviceread204)
    ;DEVICE(d0, 0, deviceinit206, deviceread206)
    %DEVICE(w0, 0, deviceinit215, deviceread215)
    %DEVICE(wf0, 8, deviceinit215, deviceread215)
    %DEVICE(wf1, 9, deviceinit215, deviceread215)
    %DEVICE(wf2, 10, deviceinit215, deviceread215)
    %DEVICE(wf3, 11, deviceinit215, deviceread215)
    ;DEVICE(b0, 0, deviceinit254, deviceread254)
%END_BOOTSTRAP
```

MONITOR CONFIGURATION SUBMIT FILES

Each supported iSBC board has several submit files which will perform various configuration options and are included on the diskettes in the iSBC 957B package. The configuration options are device changes, NPX support and the Bootstrap Loader option. Each submit file will also burn the configured monitor code into 2732 EPROMs. The files required to be on Drive 0 for the monitor configuration submit files are:

```
ASM86.86
LINK86.86
LOC86.86
OH86
UPM  V3.2 or newer
DELETE
SUBMIT
```

The nine monitor configuration submit files provided in the iSBC 957B package are listed in Table 6-2. A SERIES-III Intellec development system is required to run any of these submit files. A four drive floppy diskette-based Intellec development system is required to run any of these submit files. Additionally, a two platter hard disk-based Intellec development system may be used to run all the configuration submit files except those that configure the Bootstrap Loader. The Bootstrap Loader configuration submit files, as supplied, require the existence of a drive 2. Any of the drive specifications in the submit files may be reassigned to suit your requirements.

Table 6-2. iAPX 86, 88 Monitor Configuration Submit Files

	Default and Device Changes	NPX Support	Bootstrap Loader Option
iSBC 86/12A	CF8612.CSD	CF8612.CSD	CB8612.CSD
iSBC 86/05	CF8605.CSD	CF8605.CSD	CB8605.CSD
iSBC 88/40	CF8840.CSD	CX8840.CSD	CB8840.CSD
iSBC 88/25	CF8825.CSD	CF8825.CSD	CB8825.CSD

DEFAULT AND DEVICE CHANGE SUBMIT FILES

The four submit files in the category "Default and Device Changes" all have the format "CFxxxx.CSD. These will assemble the configuration source module, link it with the rest of the monitor code, locate that configuration appropriately and then will burn the located code into 2732 EPROMs. To use these submit files, place the single or double density diskette, according to your hardware, provided in the iSBC 957B package, into drive 1. These submit files, as supplied, expect all source modules and $INCLUDE files to be on the diskette in drive 1. Next modify the desired configuration source module (if necessary) to meet your requirements and then type:

```
SUBMIT :F1:CFxxxx(date)
```

where

xxxx — are four numbers which describe the single board computer used. See Table 6-2.

date — is the date. Up to 9 characters may be used

The final action of these submit files is to burn the monitor into 2732 EPROMs. Attach a Universal PROM Programmer (UPP) to the UPP port on your Intellec development system and then follow the directions printed on the Intellec console. The submit file will request a control-E (E) before starting to burn the EPROMs.

An example, using a iSBC 86/05 configuration file, of the SERIES-III ASM86 command which assembles a configuration source file of the iAPX 86, 88 monitor is given below.

```
RUN ASM86 (:f1:cf8605.a86) print(:f1:cf8605.lst) date(%0) errorprint
```

An example, using a iSBC 86/05 configuration file, of the SERIES-III LINK86 command which links a configuration of the iAPX 86, 88 monitor is given below.

```
RUN LINK86 &
    :F1:montor.lib(montor), &
    :F1:cf8605.obj, &
    :F1:montor.lib, &
    8087.lib &
    to :F1:m8605.lnk  print(:F1:m8605.mp1)
```

An example, using a iSBC 86/05 configuration file, of the SERIES-III LOC86 command which locates a configuration of the iAPX 86, 88 monitor is given below.

```
RUN LOC86 :f1:m8605.lnk &
    to :f1:m8605 print(:f1:m8605.mp2) map &
    addresses(segments(data(0h),code(0FC000h),monvect(0FFFB0h))) &
    order(segments(data,stack,code)) &
    noinitcode
```

All default configurations are located at 0FC000H except the configuration for the iSBC 88/40 board. The default configuration for the iSBC 88/40 board locates the monitor at 0FD000H to allow the use of a 2816 EEPROM at 0FC000H. Any configuration of the monitor which does not include NPX support may be located at 0FD000H.

The DATA segment must be located at 0h to allow the monitor to correctly set up the hardware and software interrupt vectors. If the DATA segment is not located at 0h, application code or data could corrupt the monitor's data.

The MONVECT segment must be located at 0FFFB0h to allow the monitor to be reset correctly and to allow the System I/O routines to function.

An example of the OH86 and UPM commands that burn a configuration of the iAPX 86, 88 monitor for the iSBC 86/05 board into 2732 EPROMs is given below.

```
;
OH86 :f1:m8605 to :f1:m8605.hex
;
UPM
2732
socket=2
read 86hex file :f1:m8605.hex from 0 to 03FFFH start 0FC000H
strip low from 0 to 03FFFH into 4000H
strip high from 0 to 03FFFH into 6000H
program from 4000H to 04FFFH start 0
program from 5000H to 05FFFH start 0
program from 6000H to 06FFFH start 0
program from 7000H to 07FFFH start 0
exit
```

NPX SUPPORT SUBMIT FILES

NPX support is included in all the default (Intel-supplied) configurations of the monitor except the configuration for the iSBC 88/40 board. Hence all default and device change submit files may optionally configure NPX support into or out of the monitor except the default submit file for the iSBC 88/40 board. If NPX support is to be configured into the monitor to run on the iSBC 88/40 board, the submit file, CX8840.CSD, must be used.

To use CX8840.CSD, attach a UPP to the UPP port on the Intellec development system and type:

```
SUBMIT :F1:CX8840(date,cnf_file)
```

where

date	is the date. Up to 9 characters may be used.
cnf_file	is the name of the configuration source module which specifies the NPX support option for the iSBC 88/40 board. An example of the format of this name is "CX8840". The name extension ".A86" and the file's existence on drive 1 are assumed by CX8840.CSD. This name may be from 1 to 6 characters long.

The final action of this submit file is to burn the monitor with NPX Support configured in into four 2732s. Follow the directions printed on the Intellec console.

BOOTSTRAP LOADER OPTION CONFIGURATION SUBMIT FILES

The iSBC 957B package provides you the capability to configure the iRMX 86 or 88 Bootstrap Loader first stage into the monitor. For this discussion, it will be assumed that you are familiar with the Bootstrap Loader. When the Bootstrap Loader option is selected, that is, the Bootstrap Loader first stage is configured into the monitor, the "B" command becomes a valid monitor command, causing the Bootstrap Loader to load the specified code when entered. If the Bootstrap Loader option is not selected, the "B" command is invalid. No default monitor configuration selects the Bootstrap Loader option.

Selecting the Bootstrap Loader option is performed by modifying the iAPX 86, 88 Monitor-Bootstrap Loader Configuration Table in the desired monitor configuration source file. To select the Bootstrap Loader option, replace the ";" with a "%" in front of the BOOTSTRAP and END_BOOTSTRAP macros. Next, determine which devices you wish the Bootstrap Loader to load from and replace the ";" with a "%" in front of the DEVICE macros that describe those devices. If necessary, modify the parameters in the appropriate macro calls to reflect the attributes of the devices in your system. iRMX 86 user's should see the iRMX 86 CONFIGURATION GUIDE for details.

If you are a user of the iRMX 88 Interactive Configuration Utility (ICU), the first stage device driver configuration source files for the storage devices in your system are generated automatically for you by running the ICU. In order to make these source files compatible with the monitor Bootstrap Loader configuration submit files, the physical location of all source and include files specified to the ICU must be drive 2. Also, ICU users must copy the Bootstrap Loader library, BS1.LIB, to the ICU configuration diskette when the ICU is finished. If you are not a user of the iRMX 88 ICU, you must optionally modify the first stage device driver configuration source files, contained on the Bootstrap Loader Release diskette, to meet your device specifications.

Next, optionally modify the appropriate Intel-supplied Bootstrap Loader option configuration submit file as described here. These submit files, used to configure the Bootstrap Loader first stage into the monitor, are all of the format "CBxxxx.CSD (see Table 6-2). The Bootstrap Loader option configuration submit files, as supplied on the diskettes in the iSBC 957B package, will configure support for the iSBC 204 Flexible Diskette Controller and the iSBC 215 Winchester Disk Controller into the monitor via the Intel-supplied first stage device drivers. However, any Intel-supplied or custom first stage device drivers may be configured into the monitor.

If these two device drivers meet your device requirements, no changes need to be made to the Intel-supplied Bootstrap Loader option configuration submit files. Place either the Bootstrap Loader Release diskette or the iRMX 88 system configuration diskette generated by the iRMX 88 ICU into drive 2 of your Intellec development system. iRMX 88 ICU users must also copy BS1.LIB to the ICU generated diskette. To invoke a Bootstrap Loader option configuration submit file, type:

```
SUBMIT :F1:CB8612(date, cnf_file, bootstrap_address)
```

where

date	is the date of configuration. Up to 9 characters may be used.
cnf_file	is the name of the monitor configuration source module which contains both device specifications and selection of the Bootstrap Loader option. An example of the format of this name is "CB8612". This file is assumed to be on drive 1 and its name is assumed to have the name extension ".A86". This name may be from 1 to 6 characters long.
bootstrap_address	is the address the second stage Bootstrap Loader code will be read into. The default base is decimal. An explicit "H" must be used to specify a hexadecimal address.

The default submit file to configure the monitor with the Bootstrap Loader option included for the iSBC 86/12A board is given below as an example. All Bootstrap Loader option configuration submit files, as supplied, select the same two devices.

```
;
; *-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
;
;        iAPX 86, 88 MONITOR - BOOTSTRAP LOADER DEVICE DRIVER
;                          SELECTION TABLE
;
;   Enter control E to begin.
RUN ASM86 :f2:b204.a86 object(:f1:b204.obj) print(:f2:b204.lst)
        workfiles(:f1:,:f1:) date(%0)
;RUN ASM86 :f2:b206.a86 object(:f1:b206.obj) print(:f2:b206.lst)
        workfiles(:f1:,:f1:) date(%0)
RUN ASM86 :f2:b215.a86 object(:f1:b215.obj) print(:f2:b215.lst)
        workfiles(:f1:,:f1:) date(%0)
;RUN ASM86 :f2:b254.a86 object(:f1:b254.obj) print(:f2:b254.lst)
        workfiles(:f1:,:f1:) date(%0)
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *-*
;
RUN ASM86 :f1:%1.a86 print(:f1:%1.lst) date(%0) errorprint
;
RUN LINK86 &
    :f1:montor.lib(montor), &
    :f1:%1.obj, &
    :f1:montor.lib, &
    :f1:b204.obj, &
    :f1:b215.obj, &
    :f2:bs1.lib, &
    8087.lib &
    to :f1:mb8612.lnk print(:f1:mb8612.mp1)
;
RUN LOC86 :f1:mb8612.lnk addresses(segments(data(0H), &
    data_bs204(%2), &
    code(0FC000H),monvect(0FFFB0H))) &
    order(segments(data,stack, &
    data_bs204, &
    data_bs215, &
    bootstack,boot,code, &
    code_bs204, &
    code_bs215 &
    )) &
    segsize(boot(1800H)) &
    noinitcode map print(:f1:mb8612.mp2)
```

If your device requirements are not met by the first stage device drivers configured by the Intel-supplied submit files, modifications must be made to the submit files. Up to three changes, to different sections of the appropriate Bootstrap Loader option submit file, must be made in order to change that submit file to configure the Bootstrap Loader to load code from different devices than are configured by the default Bootstrap Loader submit files.

1) Determine which devices you require. Modify the Device Driver Selection Table in the appropriate configuration submit file to assemble the first stage device driver configuration files for the desired devices.

2) Modify the LINK86 command to reflect the device driver selection changes

3) Modify the LOC86 command to reflect the same device driver selection changes.

Below is an example, selecting the iSBC 206 Disk Controller and the iSBC 254 Bubble Memory Controller as the first stage device drivers to be configured into the monitor and from which the Bootstrap Loader may read. The required changes to the default Bootstrap Loader option configuration submit file are marked with arrows. The changes you make to the submit file must constitute a valid LINK86 or LOC86 command.

```
    ; *-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
    ;
    ;        iAPX 86, 88 MONITOR - BOOTSTRAP LOADER DEVICE DRIVER
    ;                          SELECTION TABLE
    ;
--> ;RUN ASM86 :f2:b204.a86 object(:f1:b204.obj) print(:f2:b204.lst)
            workfiles(:f1:,:f1:) date(%0)
--> RUN ASM86 :f2:b206.a86 object(:f1:b206.obj) print(:f2:b206.lst)
            workfiles(:f1:,:f1:) date(%0)
--> ;RUN ASM86 :f2:b215.a86 object(:f1:b215.obj) print(:f2:b215.lst)
            workfiles(:f1:,:f1:) date(%0)
--> RUN ASM86 :f2:b254.a86 objedt(:f1:b254.obj) print(:f2:b254.lst)
            workfiles(:f1:,:f1:) date(%0)
    ;
    ; *-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
    ;
    RUN ASM86 :f1:%1.a86 print(:f1:%1.lst) date(%0) errorprint
    ;
    RUN LINK86 &
        :f1:montor.lib(montor), &
        :f1:%1.obj, &
        :f1:montor.lib, &
    --> :f1:b206.obj, &
    --> :f1:b254.obj, &
        :f2:bsl.lib, &
        8087.lib &
        to :f1:mb8612.lnk print(:f1:mb8612.mpl)
    ;
    RUN LOC86 :f1:mb8612.lnk addresses(segments(data(0H), &
    --> data_bs206(%2), &
        code(0FC000H),monvect(0FFFB0H))) &
        order(segments(data,stack, &
    --> data_bs206, &
    --> data_bs254, &
        bootstack,boot,code, &
    --> code_bs206, &
    --> code_bs254 &
        )) &
        segsize(boot(1800H)) &
        noinitcode map print(:f1:mb8612.mp2)
```

The location the second stage bootstrap code is read into is determined by the line "data_bs206(%2)" where "%2" will receive the bootstrap address parameter from the SUBMIT invocation. The segment listed on this line must also be the first data_bsxxx segment listed in the ORDER control statement.

BOTH NPX SUPPORT AND BOOTSTRAP LOADER SUPPORT FOR THE iSBC 88/40 BOARD

No Intel-supplied configuration submit file will generate a monitor with both NPX support and the Bootstrap Loader option configured in for the iSBC 88/40 board. If such a monitor is to be configured to run on the iSBC 88/40 board, the submit file, "CB8840.CSD", must be modified to perform the configuration and to burn the monitor into EPROM. Make the following four changes:

1) Change the address used to locate the code, 0FD000h, in the line
   "code(0FD000h),monvect(0FFFB0h))) &"

   to the address, 0FC000h, as follows
   "code(0FC000h),monvect(0FFFB0h))) &"

2) Change the line
   "read 86hex file :f1:%1.hex from 0 to 02FFFh start 0FD000h"

   to
   "read 86hex file :f1:%1.hex from 0 to 03FFFh start 0FC000h"

3) After the line
   "(control-E)program from 02000h to 02FFFh start 0h"

   add the following line
   "(control-E)program from 03000h to 03FFFh start 0h"

4) After the line
   "; following order:"

   add the line
   "; U76"

The first EPROM burned by the revised submit file will be inserted into socket U76 on the iSBC 88/40.

Now run this submit file as any other Bootstrap Loader option configuration submit file, making certain to specify NPX support in the monitor configuration source file used.

APPENDIX A. NUMBERED ISIS-II ERROR MESSAGES

This appendix lists the numbered error messages issued by the various ISIS-like commands. By convention, error numbers 1-99 inclusive are reserved for errors that originate in or are detected by the resident routines of ISIS-II; error numbers 100-199 inclusive are reserved for user programs; and numbers 200-255 inclusive are used for errors that may be encountered by nonresident system routines. In the following list an asterisk precedes error numbers that are always fatal. The other errors are generally nonfatal unless they are issued by the CONSOL system call. See Tables 1 and 2 below.

0	No error detected.
*1	Insufficient space in buffer area for a required buffer.
2	AFTN does not specify an open file.
3	Attempt to open more than 6 files simultaneously.
4	Illegal filename specification.
5	Illegal or unrecognized device specification in filename.
6	Attempt to write to a file open for input.
*7	Operation aborted; insufficient diskette space.
8	Attempt to read from a file open for output.
9	No more room in diskette directory.
10	Filenames do not specify the same diskette.
11	Cannot rename a file; name already in use.
12	Attempt to open a file already open.
13	No such file.
14	Attempt to open for writing (output or update) or to delete or rename a write-protected file.
*15	Attempt to load into ISIS-II area or buffer area.
16	Checksum error.
17	Attempt to rename or delete a file not on diskette.
*18	Unrecognized system call.
19	Attempt to seek in a file not on diskette.
20	Attempt to seek backward past beginning of file.
21	Attempt to rescan a file not line edited.
22	Illegal ACCESS parameter to OPEN or access mode impossible for file specified (input mode for :LP:, for example).
23	No filename specified for a diskette file.
*24	Input/output error on diskette.
25	Incorrect specification of echo file to OPEN.
26	Incorrect ATTRIBUTE parameter in ATTRIB system call.
27	Incorrect MODE parameter in SEEK system call.
28	Null file extension.
*29	End of file on console input.
*30	Drive not ready.
31	Attempted seek on file open for output.
32	Can't delete an open file.
*33	Illegal system call parameter.
34	Bad SWITCH parameter to LOAD.
35	Attempt to extend a file opened for input by seeking past end-of-file.
201	Unrecognized switch.
202	Unrecognized delimiter character.

203 Invalid command syntax.
204 Mature end of file.
206 Illegal diskette label.
207 No END statement found in input.
208 Checksum error.
209 Illegal record sequence in object module file.
210 Insufficient memory to complete job.
211 Object module record too long.
212 Bad object module record type.
213 Illegal fixup specified in object module file.
214 Bad parameter in a SUBMIT file.
215 Argument too long in a SUBMIT file.
216 Too many parameters is a SUBMIT file.
217 Object module record too short.
218 Illegal object module record format.
219 Phase error.
220 No end of file record in object module file.
221 Segment exceeds 64K bytes.
222 Unrecognized record in object module file.
223 Fixup record pointer is incorrect.
224 Illegal record sequence in object module file.
225 Illegal module name specified.
226 Module name exceeds 31 characters.
227 Command syntax requires left parenthesis.
228 Command syntax requires right parenthesis.
229 Unrecognized control specified in command.
230 Duplicate symbol found.
231 File already exists.
232 Unrecognized command.
233 Command syntax requires a "TO" clause.
234 File name illegally duplicated in command.
235 File specified in command is not a library file.
236 More than 249 common segments in input files.
237 Specified common segment not found in object file.
238 Illegal stack content record in object file.
239 No module header in input object file.
240 Program exceeds 64K bytes.

When error number 24 occurs, an additional message is output to the console.

FDCC = 00nn, DRIVE = mm

where nn has the following meanings:

01 Deleted record.
02 CRC error (data field).
03 Invalid address mark.
04 Seek error.
08 Address error.
0A CRC error (ID field).
0E No address mark.
0F Incorrect data address mark.
10 Data overrun or data underrun.
20 Write protect.
40 Write error.
80 Not ready.

Table A-1. Nonfatal Error Numbers Returned by System Calls

OPEN	3,4,5,9,12,13,14,22,23,25,28
READ	2,8
WRITE	2,6
SEEK	2,19,20,27,31,35
RESCAN	2,21
CLOSE	2
DELETE	4,5,13,14,17,23,28,32
RENAME	4,5,10,11,13,17,23,28
ATTRIB	4,5,13,23,26,28
ERROR	None.
LOAD	3,4,5,12,13,22,23,28,34
EXIT	None.

Table A-2. Fatal Errors Issued by System Calls

OPEN	1,7,24,30,33
READ	24,30,33
WRITE	7,24,30,33
SEEK	7,24,30,33
RESCAN	33
CLOSE	33
DELETE	1,24,30,33
RENAME	1,24,30,33
ATTRIB	1,24,30,33
ERROR	33
LOAD	1,15,16,24,30,33

APPENDIX B. iAPX 86, 88 MONITOR ERROR MESSAGES

This appendix lists the error messages issued by the various iAPX 86, 88 monitor commands.

Syntax Error.
Bad Command.
Invalid Expression.
Bad EMDS Connection.
Improper nesting of repeat factors.
Cannot write in location.
INT 1 while not single stepping.
Divide Error Occurred.
Overflow occurred.
NPX Unavailable.
Bad Patch Byte.

APPENDIX C. EQUIPMENT SUPPLIED WITH THE iSBC 957B™ PACKAGE

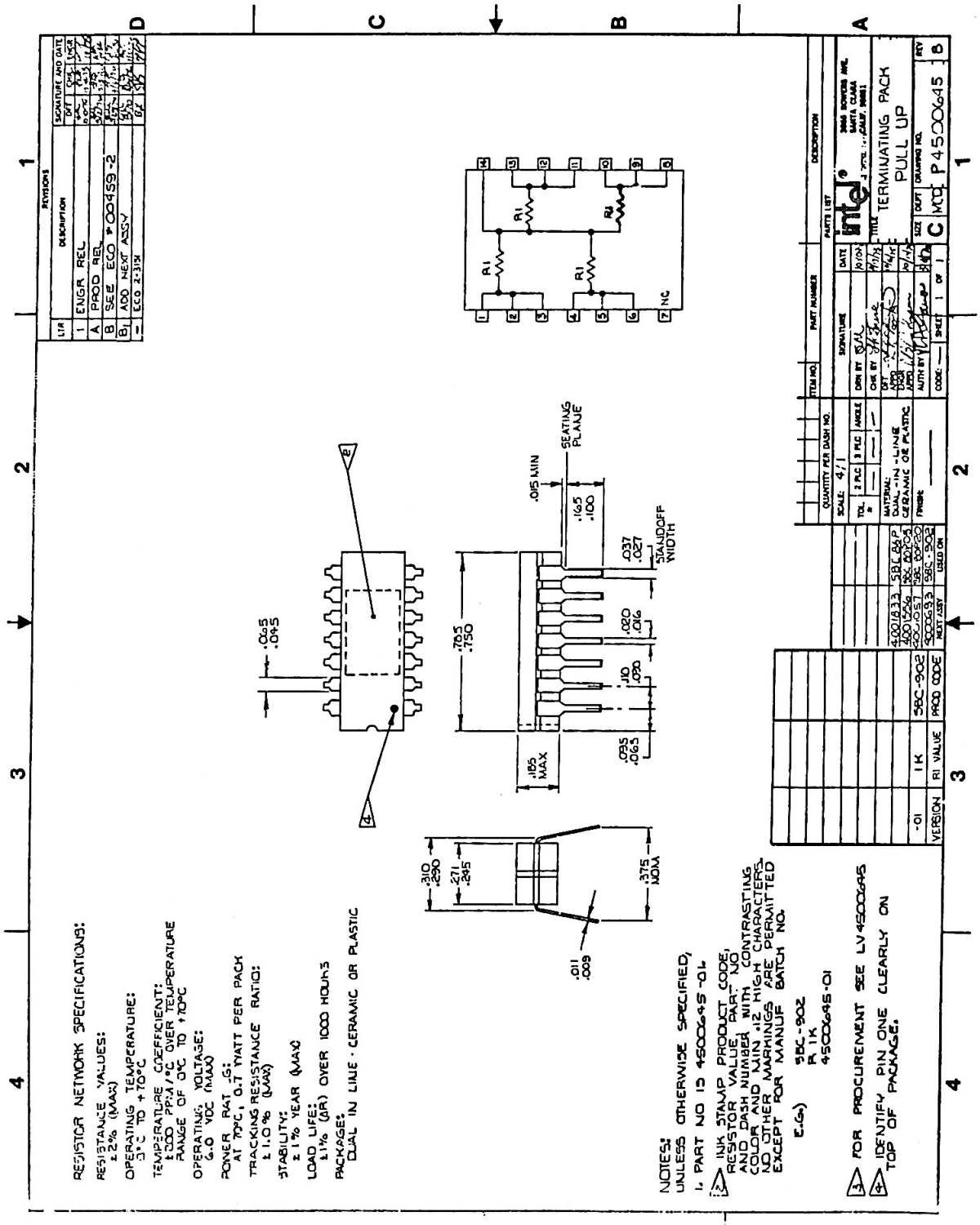
Equipment Supplied

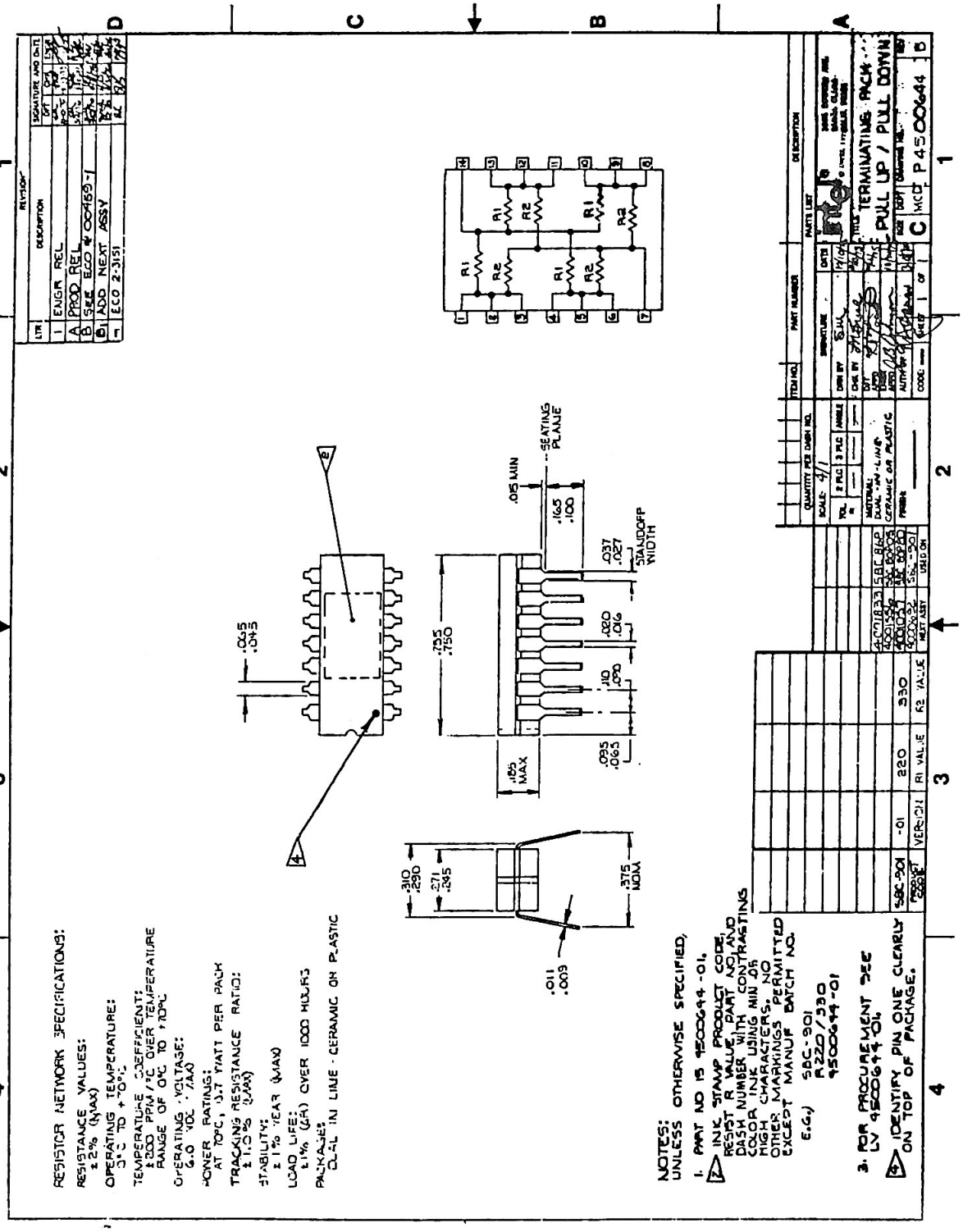
Part Number	Qty.	Description
142823-01	1	Single Density Diskette containing ISIS-II iAPX 86, 88 loader.
142822-01	1	Double Density Diskette containing ISIS-II iAPX 86, 88 loader.
143780-001	1	I.C., Intel 2732 EPROM (Monitor Program).
143781-001	1	I.C., Intel 2732 EPROM (Monitor Program).
143782-001	1	I.C., Intel 2732 EPROM (Monitor Program).
143783-001	1	I.C., Intel 2732 EPROM (Monitor Program).
143784-001	1	I.C., Intel 2732 EPROM (Monitor Program).
143785-001	1	I.C., Intel 2732 EPROM (Monitor Program).
143786-001	1	I.C., Intel 2732 EPROM (Monitor Program).
1002129	1	Status Adapter (printed circuit board with 14 pins arranged to plug into integrated circuit socket).
4500644	4**	iSBC 901 Resistor Pack, Pull up/Pull down.
4500645	4**	iSBC 902 Resistor Pack, Pull up.
54-068	4	I.C., 7437, quadruple 2-input positive-NAND buffers.
4002127	1*	Cable Assembly, RS232C Up/Down Load (round cable with 25-pin male connector at each end).
4002287	1	Cable Assembly, Parallel Up/Down Load (flat cable with 50-pin edge connector at one end and an adapter to 25-pin male connector at other end).
400677	1	Cable Assembly, OEM RS232C Input/Output (flat cable with 26-pin edge connector at one end and 25-pin RS232C connector at other end).
84-009	8	Screw, panhead, 4-40 x 0.25 inch.
143979	1	User's Guide for the iSBC 957B, iAPX 86, 88 Interface and Execution Package

NOTE: iSBC 901 Resistor Packs and 7437 IC's are not used in the implementation of the interface and execution package. These parts are supplied to use at your discretion.

* Assembly drawings supplied with cable assemblies.
** Schematic drawings for resistor packs, see appendix D.

APPENDIX D. iSBC 901™ AND iSBC 902™ SCHEMATICS




REVISIONS
LTR DESCRIPTION SIGNATURE AND DATE
1 ENGR REL
A PROD REL
B SEE ECO # C04169-1
C ADD NEXT ASSY
D ECO 2-3151
RESISTOR NETWORK SPECIFICATIONS:
RESISTANCE VALUES:
±2% (MAX)
OPERATING TEMPERATURE:
0°C TO +70°C
TEMPERATURE COEFFICIENT:
±200 PPM/°C OVER TEMPERATURE
RANGE OF 0°C TO +70°C
OPERATING VOLTAGE:
6.0 VDC (MAX)
POWER RATING:
AT 70°C, 0.7 WATT PER PACK
TRACKING RESISTANCE RATIO:
±1.0% (MAX)
STABILITY:
±1% YEAR (MAX)
LOAD LIFE:
±1% (ΔR) OVER 1000 HOURS
PACKAGE:
DUAL IN LINE - CERAMIC OR PLASTIC
.065 .045
.755 .750
.095 MIN
SEATING PLANE
.165 .100
.037 .027
STANDOFF WIDTH
.020 .016
.110 .090
.095 .065
.185 MAX
.310 .290
.271 .245
.375 MAX
.011 .009
R1 R2 R3
NOTES:
UNLESS OTHERWISE SPECIFIED,
1. PART NO IS 9500644-01.
2. INK STAMP PRODUCT CODE, RESIST R VALUE, PART NO AND DASH NUMBER WITH CONTRASTING COLOR INK USING MIN .05 HIGH CHARACTERS. NO OTHER MARKINGS PERMITTED EXCEPT MANUF BATCH NO.
E.G.) SBC-901 R220/330 9500644-01
3. FOR PROCUREMENT SEE LV 9500644-01.
4. IDENTIFY PIN ONE CLEARLY ON TOP OF PACKAGE.
PARTS LIST
ITEM NO. PART NUMBER DESCRIPTION
QUANTITY PER DASH NO.
SCALE: 4/1
TOL. 2 PLC 3 PLC ANGLE
DRAWN SJH 12/6/78
CHK BY
ENGR
MFG
QUALITY ASSUR
MATERIAL DUAL-IN-LINE CERAMIC OR PLASTIC
FINISH
NEXT ASSY USED ON
SBC-901 -01 220 330
PRODUCT VERSION R1 VALUE R2 VALUE
TERMINATING PACK
PULL UP / PULL DOWN
SIZE C MCD P/N 9500644
DRAWING NO.
SHEET 1 OF 1
CODE —

8087 (See NPX)
8251A (See USART)
8253 (See Programmable Interval Timer)
8254 (See Programmable Interval Timer)
8255 (See Programmable Peripheral Interface)
8259A (See Programmable Interrupt Controller)

Address Specification 3-7
APXLOD 3-1

BAUD_RATE Macro (See Configuration Macros)
BAUD_RATE_TIMER Macro (See Configuration Macros)
Baud Rates 3-1, 6-4, 6-5
BCD Numbers (See NPX Data Types)
BOOTSTRAP Macro (See Configuration Macros)
Bootstrap Loader Option
    Configuration Submit Files 6-16
    Default Configuration of Bootstrap Loader 6-17
    Modification of Default Configuration 6-18
    Sample Configuration Source File 6-13
Burning the iAPX 86, 88 Monitor into 2732 EPROMs 6-15

CF975B.MAC 6-1
Command Descriptions
    Summary 3-11
    Bootstrap (B) 3-33, 6-11, 6-13, 6-16
    Comment (*) 3-32
    Compare Memory (C) 3-29
    Display Memory (D) 3-22
    Examine/Modify Registers (X) 3-18
    Exit (E) 3-33
    Find Memory (F) 3-28
    Go (G) 3-14
    Load (L) 3-13
    Load and Go (R) 3-15
    Move Memory (M) 3-27
    Single Step (N) 3-17, 5-2
    Substitute Memory (S) 3-25
    Port Input (I) 3-29
    Port Output (O) 3-30
    Print String (P) 3-31
    Upload (T) 3-16
Command Line
    Carriage Return 3-10
    Control-C 3-10
    Control-Q 3-10
    Control-R 3-10
    Control-S 3-10
    Control-X 3-10
    Entering 3-10
    Errors 3-9

Command Structure Definition 3-2
    Byte and Word Variables 3-2
    Numeric Variables 3-3
Common Bus Request Option
    iSBC 86/05 2-4
Configuration Macros 6-2
    BAUD_RATE macro 6-5
    BAUD_RATE_TIMER macro 6-6
    BOOTSTRAP macro 6-11
    CPU macro 6-3
    DEVICE macro 6-11
    END_BOOTSTRAP macro 6-12
    EXTRA_TIMER macro 6-7
    INTERRUPT_CONTROLLER macro 6-9
    Invalid Parameters in macro calls 6-3
    MAX_BAUD_RATE_COUNT macro 6-4
    NPX macro 6-10
    PARALLEL_PORT macro 6-9
    SERIAL_PORT macro 6-8
Configuration Source Files 6-1
    Assembly of, 6-15
Configuration Submit Files 6-14
    Equipment and Software required to use 6-13
Configuring the iAPX 86, 88 Monitor 6-1
Continuation Factor 3-7
CPU macro (See Configuration Macros)

DEVICE macro (See Configuration Macros)
Disassembly (See Command Descriptions - Display Memory)

EPROM
    Type Required for Monitor 6-13
    Burning with the Monitor 6-15
EEPROM Option 1-1
    iSBC 88/40 2-5, 6-7
    Programming Pulse Generation 5-5
END_BOOTSTRAP macro (See Configuration Macros)
EXTRA_TIMER macro (See Configuration Macros)

iAPX 86, 88 CPU Registers 3-8, 5-3
ICE 86 1-1
Installing Monitor EPROMs 2-7
Instruction Breakpoints 5-5
Integer Numbers (See NPX Data Types)
Interfacing Without an Intellec System 2-14, 3-1
INTERRUPT_CONTROLLER macro (See Configuration Macros)
iSBC 901 Resistor Packs 1-2, App. D
iSBC 902 Resistor Packs 2-11, App. D
iSBC 957A Intellec-iSBC 957A Interface and Execution Package 1-1
iSBX 351 Serial Multimodule 2-4, 6-7

Linking an iAPX 86, 88 Monitor Configuration 6-15, 6-18, 6-19
Loader Commands 3-1
Locating an iAPX 86, 88 Monitor Configuration 6-15, 6-18, 6-19

MAX_BUAD_RATE_COUNT macro (See Configuration Macros)
Memory Organization 5-1, 6-15, 6-20
Multiple Commands on a Line 3-7

Non-maskable Interrupt 2-1
    iSBC 86/05 2-4
NPX
    Data Types 3-3, 3-4, 3-10, 3-22, 3-26
    Data Display Formats 3-5
    Range of Data Types 3-4
    Memory used by NPX Support 5-1, 6-10
    Number Suffixes 3-4
    Numeric Processor Extension 1-1
    Registers 3-9
    Scientific Number Format 3-5
    Support 3-3, 6-10, 6-16
NPX macro (See Configuration Macros)
Numeric Processor Extension (See NPX)

Parallel Interfacing Cabling
    Intellec Series II Model 210 2-12
    Intellec Series II Model 22-/230 2-13
    Intellec 800 2-13
Parallel Interfacing Processor Board Modifications
    Add Jumpers 2-11
    Default Jumpers 2-11
    iSBC 902 Resistor Packs 2-11
    Remove Jumpers 2-11
    Status Adapter 2-11
Parallel Interfacing Requirements 2-11
PARALLEL_PORT macro (See Configuration Macros)
Programmable Interrupt Controller (PIC) 5-4, 6-9, 6-13
Programmable Interval Timer (PIT) 2-2, 2-3, 2-4, 2-6 6-6, 6-7, 6-12
    Input Clock Frequency 6-4
Programmable Perpheral Interface (PPI) 2-11, 6-9, 6-12

Range of Address Specification 3-7
Real Numbers (See NPX Data Types)
Repetition Factor 3-7

Serial Interfacing Cabling
    Intellec Series II Model 210 2-9
    Intellec Series II Model 22-/230 2-10
    Intellec 800 2-10
Serial Interfacing Jumpers and Switches
    iSBC 86/12A 2-2
    iSBC 86/05 2-2
    iSBC 88/40 2-4
    iSBC 88/25 2-6
Serial Interfacing Requirements 2-1
SERIAL_PORT macro (See Configuration Macros)
Single Stepping (See Command Descriptions)

Start Up Procedure
    Entering start-up command 3-1
    Monitor Display Message 3-1
    Baud Rate Scan 3-1
System I/O Routines
    Attrib 4-11
    CI 4-14
    Close 4-8
    CO 4-14
    Delete 4-9
    Error 4-13
    Exit 4-17
    ISIS 4-15
    Load 4-12
    Open 4-2
    Read 4-4
    Rename 4-10
    Rescan 4-8
    Seek 4-6
    Write 4-5
System I/O Libraries
    APXIOS.LIB 4-2
    APXIOC.LIB 4-2
    APXIOL.LIB 4-2

Time Out Option
    iSBC 86/05 2-3
    iSBC 86/12A 2-2
    iSBC 88/25 2-6
    iSBC 88/40 2-5

USART 2-1, 5-3, 6-6, 6-8

WAIT State Selection
    iSBC 86/05 2-4
    iSBC 88/25 2-6

REQUEST FOR READER'S COMMENTS

Intel Corporation attempts to provide documents that meet the needs of all Intel product users. This form lets you participate directly in the documentation process.

Please restrict your comments to the usability, accuracy, readability, organization, and completeness of this document.

1. Please specify by page any errors you found in this manual.

2. Does the document cover the information you expected or required? Please make suggestions for improvement.

3. Is this the right type of document for your needs? Is it at the right level? What other types of documents are needed?

4. Did you have any difficulty understanding descriptions or wording? Where?

5. Please rate this document on a scale of 1 to 10 with 10 being the best rating. _____

NAME _____ DATE _____

TITLE _____

COMPANY NAME/DEPARTMENT _____

ADDRESS _____

CITY _____ STATE _____ ZIP CODE _____

Please check here if you require a written reply. ☐

WE'D LIKE YOUR COMMENTS . . .

This document is one of a series describing Intel products. Your comments on the back of this form will help us produce better manuals. Each reply will be carefully reviewed by the responsible person. All comments and suggestions become the property of Intel Corporation.

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 79 BEAVERTON, OR

POSTAGE WILL BE PAID BY ADDRESSEE

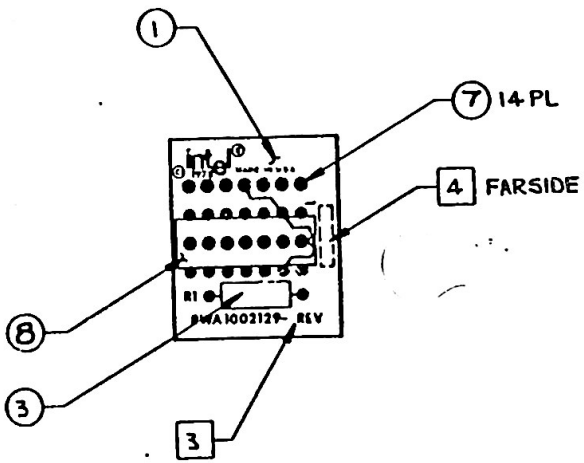
Intel Corporation
5200 N.E. Elam Young Pkwy.
Hillsboro, Oregon 97123

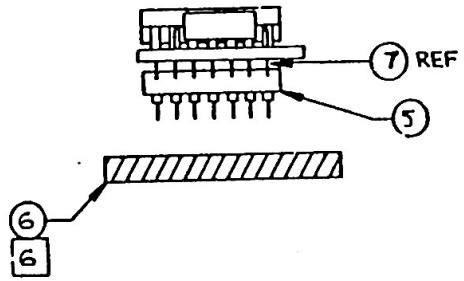
O.M.S. Technical Publications

THIS DRAWING CONTAINS INFORMATION WHICH IS THE PROPRIETARY PROPERTY OF INTEL CORPORATION. THIS DRAWING IS ACCEPTED IN CONFIDENCE AND ITS CONTENTS MUST NOT BE DISCLOSED WITHOUT THE PRIOR WRITTEN CONSENT OF INTEL CORPORATION.

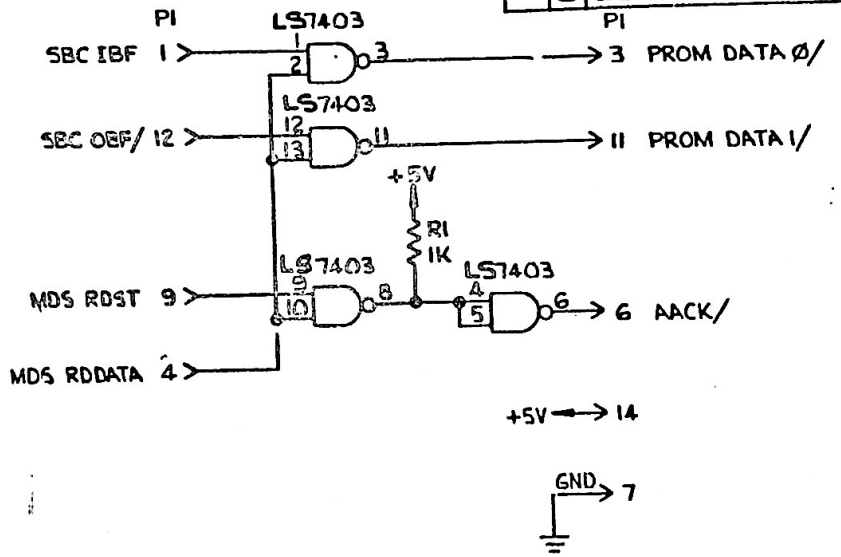
ZONE	REV	DESCRIPTION	DFT	CHK	DATE	APPROVED
		REDRAWN, ORIGINAL VELLUM LOST				
—	2	ECO 2-2724				
—	3	ECO 2-3019				
—	A	ECO 2-3151				
—	B	ECO 2-3235A				
—	—	LIMITED PRODUCTION REL	MG	CMB	1/5/79	
—	—	PRODUCTION RELEASE RWZL	AG	CMB	1/31/79	
—	C	ECO 40-2035				

D ECO 40-2347





SCHEMATIC



6 INSTALL PROTECTIVE FOAM ON SOCKET PINS.

5 REMOVED

4 MARK VENDOR I.D. WITH CONTRASTING PERM COLOR, APPROX WHERE SHOWN.

3 MARK ASSY DASH NO. AND REV LEVEL WITH CONTRASTING PERM COLOR, NON-CONDUCTIVE, .12 INCH HIGH, APPROX WHERE SHOWN.

2. WORKMANSHIP PER MCSD QAWS 99-0007-001.

1. ASSY PART NO. IS 1002129-02. ASSY AND PL ARE TRACKING DOCUMENTS.

NOTES: UNLESS OTHERWISE SPECIFIED

SEE SEPARATE PARTS LIST

ITEM NO.	PART NUMBER	DESCRIPTION
	PARTS LIST	

QUANTITY PER DASH NO.

142859 NEXT ASSY | SBC-957A USED ON

UNLESS OTHERWISE SPECIFIED:
1. DIMENSIONS ARE IN INCHES
2. BREAK ALL SHARP EDGES
3. DO NOT SCALE DRAWING
4. TOLERANCES: ANGLES ± 2°, XX ± .030, XXX ± .010
SURFACE FINISH 125

SIGNATURE	DATE
DRN BY PAHLOW	4/14/78
CHK BY	11/4/77
ENGR APVD	11/11/77
APVD	
APVD	

intel 3065 BOWERS AVE. SANTA CLARA CALIF. 95051

TITLE PRINTED WIRING ASSY STATUS ADAPTER

SIZE C | CODE OMS | DWG. NO. 1002129 | REV D

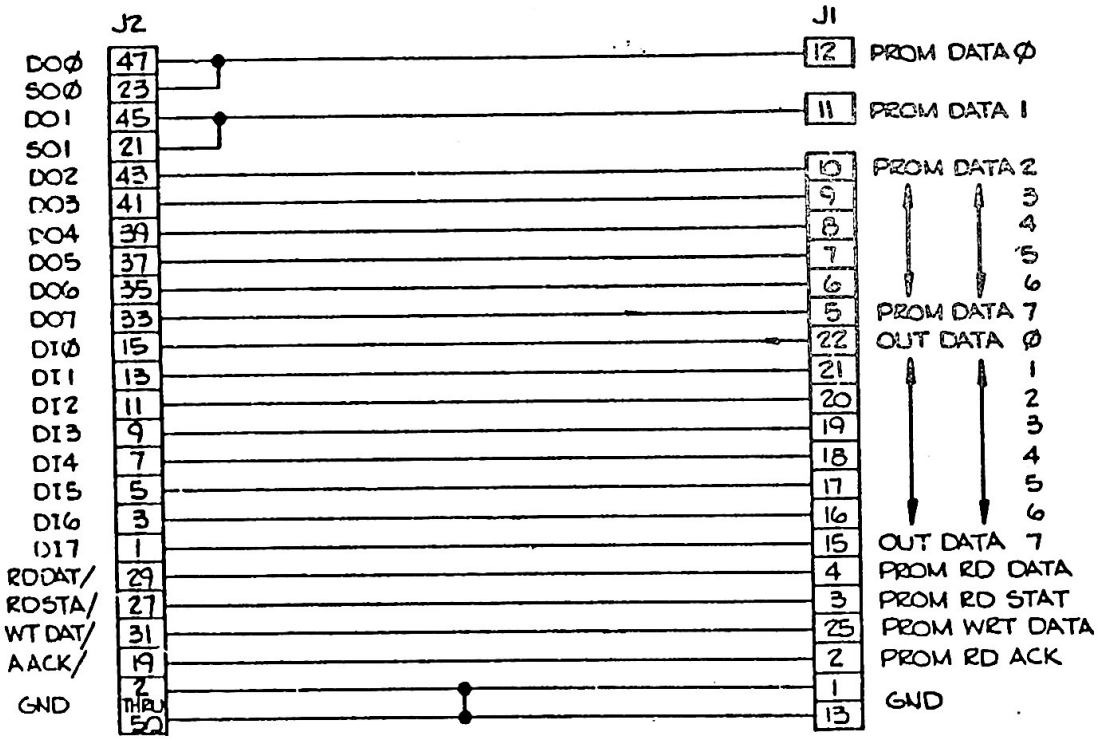
SCALE NONE | SHEET 1 OF 1

9300019

1002129

THIS DRAWING CONTAINS INFORMATION WHICH IS THE PROPRIETARY PROPERTY OF INTEL CORPORATION. THIS DRAWING IS ACCEPTED IN CONFIDENCE AND ITS CONTENTS MAY NOT BE DISCLOSED WITHOUT THE PRIOR WRITTEN CONSENT OF INTEL CORPORATION.

REVISIONS						
ZONE	REV	DESCRIPTION	DFT	CHK	DATE	APPROVED
—	1	ENGR REL	[illegible]	[illegible]	7-19-78	[illegible]
—	A	ECO 2-3151	[illegible]	[illegible]	[illegible]	[illegible]
—	B	ECO 2-3242	[illegible]	[illegible]	1/11/79	[illegible]
—	—	LIMITED PRODUCTION REL	MG	CMD	11/8/78	—
—	—	PRODUCTION REL RN16	MG	CMD	11/8/78	—



DWG. NO. P2002123 REV. B

ITEM NO.	PART NUMBER	DESCRIPTION
QUANTITY PER DASH NO.	PARTS LIST	

UNLESS OTHERWISE SPECIFIED:
1. DIMENSIONS ARE IN INCHES.
2. BREAK ALL SHARP EDGES.
3. DO NOT SCALE DRAWING.
4. TOLERANCES: ANGLES ± 2°, XX ± .030, XXX ± .010
SURFACE FINISH 125

SIGNATURE	DATE
DRN BY [illegible]	1/11/79
CHK BY [illegible]	1/11/79
ENGR APVD [illegible]	1/11/79
APVD	
APVD	

intel 3065 BOWERS AVE. SANTA CLARA CALIF. 95051

TITLE SCHEMATIC
PARALLEL LOAD ADAPTER

SIZE	CODE	DWG. NO.	REV
C	MC5D	P2002123	B

SCALE — SHEET 1 OF 1

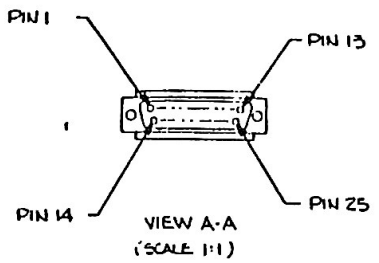
1002122 NEXT ASSY | 386-987 USED ON

THIS DRAWING CONTAINS INFORMATION WHICH IS THE PROPRIETARY PROPERTY OF INTEL CORPORATION. THIS DRAWING IS LOANED IN CONFIDENCE AND ITS CONTENTS MAY NOT BE DISCLOSED WITHOUT THE PRIOR WRITTEN CONSENT OF INTEL CORPORATION.

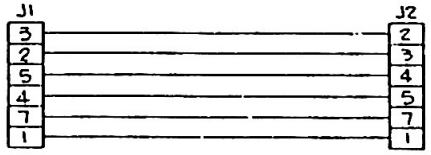
DWG NO. 4002127 SH 1 REV C

REVISIONS

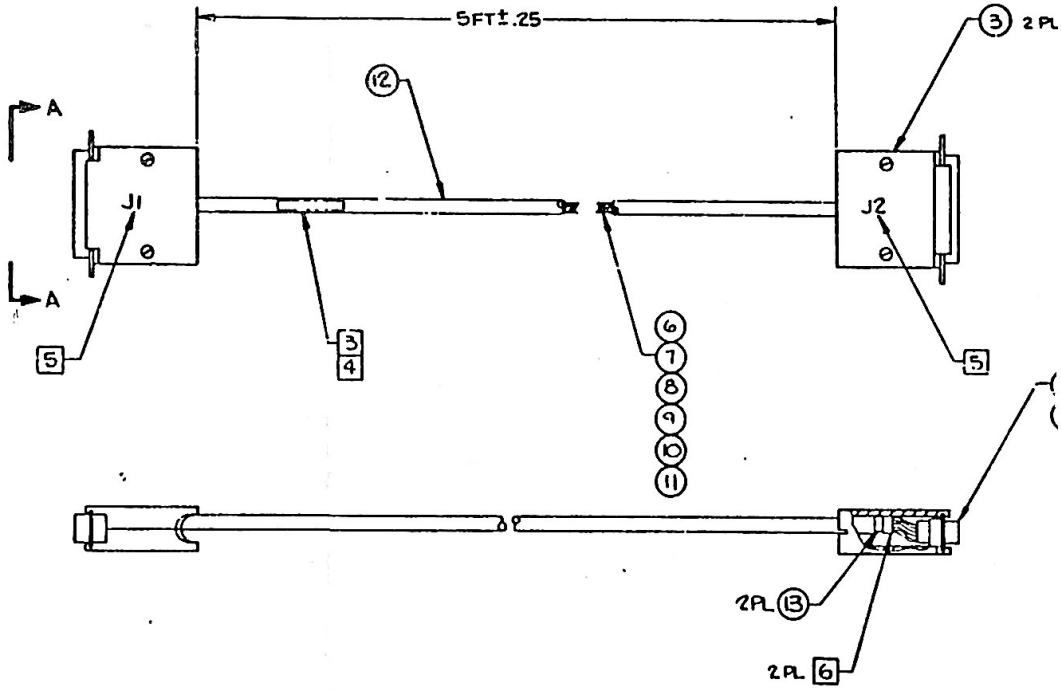
ZONE	REV	DESCRIPTION	DATE
—	1	ENGR REL.	
—	A	ECO 2-3151	
—	B	ECO 2-3243A	
—	—	LIMITED PRODUCTION RELEASE	
—	—	PRODUCTION RELEASE RWZG	
	C	ECO 10-1703	



VIEW A-A
(SCALE 1:1)



SCHEMATIC



7. WORKMANSHIP PER QAWS 99-0007-001
6. BRING INSULATION INSIDE HOOD AND BEYOND TIE WRAP APPROX 1/8 AS SHOWN
5. MARK REF. DESIG. WITH CONTRASTING PERM COLOR, APPROX WHERE SHOWN.
4. MARK ASSY VENDOR ID WITH CONTRASTING PERM COLOR, .12 HIGH, APPROX WHERE SHOWN.
3. MARK PART NO. AND REV LEVEL WITH CONTRASTING PERM COLOR, .12 HIGH, APPROX WHERE SHOWN.
2. REMOVED
1. ASSY PART NO IS 4002127-01.
ASSY WL AND PL ARE TRACKING DOCUMENTS.

NOTES: UNLESS OTHERWISE SPECIFIED

SEE SEPARATE PARTS L

ITEM NO.	PART NUMBER	QUANTITY PER DASH NO.

UNLESS OTHERWISE SPECIFIED
1. DIMENSIONS ARE IN INCHES.
2. BREAK ALL SHARP EDGES.
3. DO NOT SCALE DRAWING.
4. TOLERANCES:
ANGLES ± 2°
XX ± .030
XXX ± .010
SURFACE FINISH 7

SIGNATURE	DATE
DRN BY EJP	4/4/83
CHK BY	5/5/83
ENGR APVD	5/11/83
APVD	
APVD	

PARTS LIST

intel

TITLE
RS232 UP
CABLE, AS

SIZE	CODE	DWG.
D	0MIS	

SCALE NONE

NEXT ASSY	USED ON
4002286	SBA-957

8 7 6 5 4 3 2

INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, California 95051 (408) 987-8080

Printed in U.S.A.